REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	31.Jul.03	THESIS

4. TITLE AND SUBTITLE
"ALLOCATION OF AIR RESOURCES AGAINST AND INTELLIGENT ADVERSARY"

5. FUNDING NUMBERS

6. AUTHOR(S)
2D LT ZARYBNISKY ERIC J

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

8. PERFORMING ORGANIZATION REPORT NUMBER
CI02-1200

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
THE DEPARTMENT OF THE AIR FORCE
AFIT/CIA, BLDG 125
2950 P STREET
WPAFB OH 45433

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION AVAILABILITY STATEMENT
Unlimited distribution
In Accordance With AFI 35-205/AFIT Sup 1

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

DISTRIBUTION STATEMENT A
Approved for Public Release
Distribution Unlimited

20030822 181

14. SUBJECT TERMS	15. NUMBER OF PAGES 142
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

Allocation of Air Resources against an Intelligent Adversary

by

Eric J. Zarybnisky
B.S. Operations Research and Economics
United States Air Force Academy, 2001

SUBMITTED TO THE SLOAN SCHOOL OF MANAGEMENT IN PARTIAL FULFILLMENT OF THE REQUIREMENTS OF THE DEGREE OF

MASTER OF SCIENCE IN OPERATIONS RESEARCH
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
June 2003

Signature of Author: Eric J. Zarybnisky
Sloan School of Management
Interdepartmental Program in Operations Research
16 May, 2003

Approved by: Stephan E. Kolitz
Stephan E. Kolitz
The Charles Stark Draper Laboratory, Inc.
Technical Supervisor

Certified by: Cynthia Barnhart
Cynthia Barnhart
Professor, Civil and Environmental Engineering; and Professor, Engineering Systems Division
Co-Director, Center for Transportation and Logistics
Thesis Advisor

Certified by: Leslie P Kaelbling
Leslie P. Kaelbling
Professor, Computer Science and Engineering
Associate Director, MIT Artificial Intelligence Laboratory
Thesis Advisor

Accepted by: James B. Orlin
James B. Orlin
Edward Pennell Brooks Professor of Operations Research
Co-Director, Operations Research Center

[This Page Intentionally Left Blank]

Allocation of Air Resources against an Intelligent Adversary

by

Eric J. Zarybnisky

Submitted to the Sloan School of Management
on 16 May 2003, in Partial Fulfillment of the Requirements
for the Degree of Master of Science in Operations Research

ABSTRACT

In a battlefield situation, the use of air assets can have a large impact upon the outcome. The problem we consider is allocating scarce resources among activities that conduct pre-strike Intelligence, Surveillance, and Reconnaissance (ISR), take strike actions against, or gather battle damage assessment (BDA) information about a set of targets in order to perform the targeting cycle. We explore methods that combine Partially Observable Markov Decision Processes (POMDPs), which prescribe strike and observation policies, and integer programming formulations, which pick the optimal set of policies given resource constraints. This work adds five major contributions beyond previous work on similar problems.

The first improvement is the introduction of allocation decisions for ISR assets, which search out and identify new targets. Also included is a model of an intelligent adversary, specifically representations of regenerative and mobile targets. In addition to incorporating Cheng's Linear Support algorithm for solving two-dimensional targeting POMDPs, we incorporate the Incremental Pruning algorithm to solve higher dimensional POMDPs for target discovery and identification. Finally, we introduce a new initialization technique as well as two integer programming formulations of the targeting cycle problem. We demonstrate the computational benefits of this decomposition through a number of parameter variation tests and targeting cycle vignettes and discuss the qualitative characteristics of the solutions generated.

Thesis Supervisor:	Professor Cynthia Barnhart
Title:	Professor of Civil and Environmental Engineering Professor, Engineering Systems Division Co-Director, Center for Transportation and Logistics
Thesis Supervisor:	Professor Leslie Kaelbling
Title:	Professor of Computer Science and Engineering Associate Director, MIT Artificial Intelligence Laboratory
Technical Supervisor:	Dr. Stephan E. Kolitz
Title:	Principal Member of the Technical Staff, The Charles Stark Draper Laboratory, Inc.
Technical Advisor:	Professor Andrew P. Armacost
Title:	Assistant Professor of Operations Research Department of Management, US Air Force Academy, CO

[This Page Intentionally Left Blank]

Table of Contents

List of Figures

List of Tables

[This Page Intentionally Left Blank]

1 Introduction

Military planners have a myriad of details to consider when developing the strategy for a military campaign. Each service has special capabilities but there is some overlap in their missions. The goal of military planners is to lay out plans that use the combat resources of all the services in the best possible way. One such area of overlap is the combat air power of the Air Force, Navy, Army, and Marines. Air Operations Centers (AOC), or Combined Air Operations Centers (CAOC) as they are called if the Joint Forces Air Component Commander (JFACC) is in command, work to incorporate these capabilities by ensuring that aircraft, weapons, and sensors are used in the appropriate capacity but at the same time, not exposing the aircraft to an undue amount of risk. "An AOC is a command and control center that plans, executes and assesses aerospace operations during a contingency or conflict" [19]. AOCs and CAOCs serve as an operational facility for the Air Component Commander (ACC) to have centralized planning, direction, and control over available air resources. They are staffed with individuals from different services who work on tasks such as weather prediction, target analysis, and sortie generation.

During the intense, five-week bombing campaign that preceded the ground war in Desert Storm, our forces flew over 100,000 aircraft sorties [6]. Almost half were combat missions [35]. These combat missions supported the *military targeting process*, which finds, identifies, strikes,

and confirms destruction of enemy assets. While basic doctrine is laid out well ahead of time, the actual plans to be implemented are dependent upon the theater of operations and the current situation and are developed closer to the time of execution. During Desert Storm "...only the first 2 to 3 days of the strategic air campaign were planned in great detail, with the remainder to be based on the damage done to the high-priority targets that would be hit in the first 48 to 72 hours" [35]. It is important to develop plans that account for possible future developments and respond to a dynamic battlefield. The quicker AOCs can generate and implement plans, the better the results.

By quickly reacting to the enemy and other factors on the battlefield, we can "get inside" the enemy's *OODA loop*. The OODA loop, as shown in *Figure 1-1*, was proposed by Col. John Boyd, a US Air Force fighter pilot. Boyd proposed a fundamental decision making process for planning military operations and competing companies in the business world. Boyd recognized the importance of completing the cycle faster than your adversary, thus "getting inside" their OODA loop.

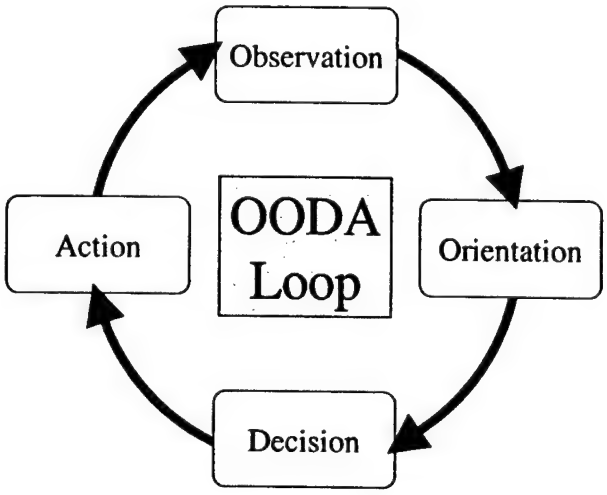

Figure 1-1: Boyd's OODA Loop

Even before the term was coined, speeding up the military OODA loop has been the focus of research. In fact, since the advent of war, militaries have been working to improve the speed at which this loop is completed. Advances such as balloon reconnaissance, the telegraph, and the telescope have all helped this effort, albeit at different levels of command. At each level of command, an OODA loop progresses in conjunction with those at the other levels of

command. It is important that all OODA loops be running smoothly because the whole organization could be slowed by a lower level unit whose OODA loop is not progressing well.

1.1 Research Scope-Targeting Cycle

An important military application of the OODA loop is found in the *targeting cycle*, which is the process of finding, identifying, destroying, and confirming the destruction of enemy assets. The targeting cycle will be explained in depth in Chapter 2. The necessity of the targeting cycle is straightforward. To carry out a successful military campaign, we must perform actions that change the will of the enemy. In some cases, this means destroying their military resources. We first need to find and identify those resources. Proper identification is becoming more important in current wars. During World War II, cities such as London and Tokyo were bombed indiscriminately. Subsequent civilian casualties were exceedingly high, which caused a backlash against this type of warfare. Since then, the US military has limited the collateral damage inflicted upon civilians by using the appropriate weapons against validated targets.

Targeting is a complicated process that exists in the overlap between intelligence and operations as shown in *Figure 1-2*.

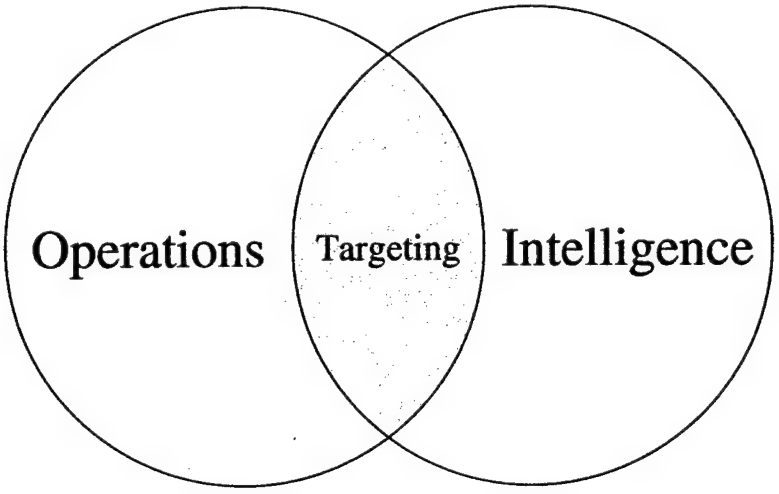

Figure 1-2: Targeting Balance between Operations and Intelligence

The operations community is responsible for the aircraft, weapons, and sensors, while the intelligence community analyzes intelligence and makes determinations about enemy capabilities. Both contribute to and benefit from effective targeting. Operators use intelligence estimates to determine appropriate routes for aircraft, and suitable sensors and weapons, to use on missions. In return they provide the intelligence community with first-hand accounts of the

battle via personal account and video footage. Using this information, and other sources, intelligence analysts can develop intelligence estimates for commanding officers, the operations community, and other end users.

This motivates the *targeting cycle problem*, which considers the allocation of resources to find, identify, destroy, and confirm the destruction of enemy resources. The targeting cycle problem we consider occurs in a dynamic, stochastic battle space with imperfect information and an intelligent adversary.

These characteristics motivate the research in this thesis and provide the framework for a solution algorithm that assigns resources to perform actions against individual objects. In the past, this has been time consuming because individuals in an AOC do it manually. An automated optimization approach can take into account far more than a human while reducing the sub-optimal effects of isolated decision-making.

1.2 Thesis Overview and Content

This thesis provides an overview of the targeting cycle and current modeling methods that have been applied to the targeting cycle problem. We introduce modeling and algorithmic changes to an existing methodology to improve the realism and computational aspects of the model and the associated solution algorithm. We enhance this approach by 1) **incorporating the discovery and identification of targets**, 2) **handling regenerative targets**, and 3) **accounting for an intelligent adversary**. These aspects of the targeting cycle problem have not been developed in earlier works and this thesis represents the first piece of work addressing these issues. In addition to a more realistic model, we also enhance the solution algorithm by proposing a **new initialization technique** as well as **two integer-programming formulations**. We run experiments based upon a basic scenario, structural variations upon that scenario, and expanded targeting vignettes. We investigate computational and planning characteristics of these solutions.

The individual chapters are summarized as follows:

Chapter 2: *The Targeting Cycle*

In this chapter we introduce the targeting cycle, the method by which the military finds, identifies, strikes, and confirms destruction of an enemy's military assets. We present a detailed description of the many stages of the targeting cycle

and the resources that are used at each stage. Interactions between the phases of the targeting cycle are discussed thus motivating a method that plans for objects, be they areas of interest, contacts, or targets, in all phases simultaneously.

Chapter 3: *Modeling the Targeting Cycle*

Modeling methods that have been applied to the targeting cycle are presented in this chapter. Two primary types of methods are described: *resource allocation* and *policy development*. We also consider a hybrid formulation that combines a resource allocation method with a policy development method. We discuss the strengths and weaknesses of these approaches in dealing with the complete targeting cycle problem. A hybrid approach using linear programming (LP) and partially observable Markov decision processes (POMDP) has been previously applied to a simplified version of the targeting cycle problem. We discuss the LP/POMDP framework, POMDP solution techniques, and develop our specific POMDP models for *object discovery*, *contact identification*, and *target destruction*.

Chapter 4: *Resource and Task Assignment*

We present a linear programming formulation that we use, along with the POMDPs, to model the allocation of resources in the targeting cycle problem. We examine variations of this basic formulation including: a new initialization procedure, alternative mixed integer programming formulations, methods for modeling regenerative targets, and a rolling horizon planning framework. Solution techniques for the LP, and its integer formulation, and POMDPs are discussed, as well as details of how information is passed between the LP and the POMDPs.

Chapter 5: *Scenarios, Results, and Analysis*

In this chapter we outline a basic targeting cycle problem scenario that we use to test the structural variations proposed in Chapter 4. We also present the results from tests on an expanded version of this scenario to assess the potential

real-time use of this technology. Results from these test are compared using solution time and benefit achieved. We also examine the solutions to understand how the model makes resource assignments and what interactions occur between plans. We discuss qualitative differences between the solution generated by the LP/POMDP hybrid approach and manually generated solutions.

Chapter 6: *Summary and Future Work*

This chapter summarizes the targeting cycle problem formulation and solution techniques along with the computational results from structural and targeting cycle vignettes. Suggested future research is also discussed.

2 The Targeting Cycle

A central focus of modern military operations is the detection, location, identification, and destruction of land based targets. As described by Joint Publication 3-60, *Joint Doctrine for Targeting* [24], the targeting process is divided into six phases that comprise the Joint Targeting Cycle, illustrated in *Figure 2-1*.

1. **Commander's Objectives, Guidance, and Intent**
2. **Target Development, Validation, Nomination, and Prioritization**
3. **Capabilities Analysis**
4. **Commander's Decision and Force Assignment**
5. **Mission Planning and Force Execution**
6. **Combat Assessment**

The focus of the first phase, *Commander's Objectives, Guidance, and Intent*, is to create "clear, quantifiable, and achievable objectives (that) lead to the successful realization of national security goals through a targeting solution" [24]. The scope of these objectives may range from wide-area campaigns affecting a large portion of the battle space to tactical level conditions. In any case, the focus of these objectives must be to change the adversary's actions so as to accomplish stated strategic goals.

Second in the targeting cycle is *Target Development, Validation, Nomination, and Prioritization,* in which the true identity of a potential target is validated. This serves two

purposes. The first is to ensure that the target is a viable part of the target set. Secondly, it must be ensured that the target is valid under the Law of Armed Conflict (LOAC). After targets are placed on the target nomination list, the *Capabilities Analysis* phase determines what resources should be used against the given targets. Factors that can influence force application include potential collateral damage to nearby facilities or noncombatants as well as the effectiveness of a weapon type against the target.

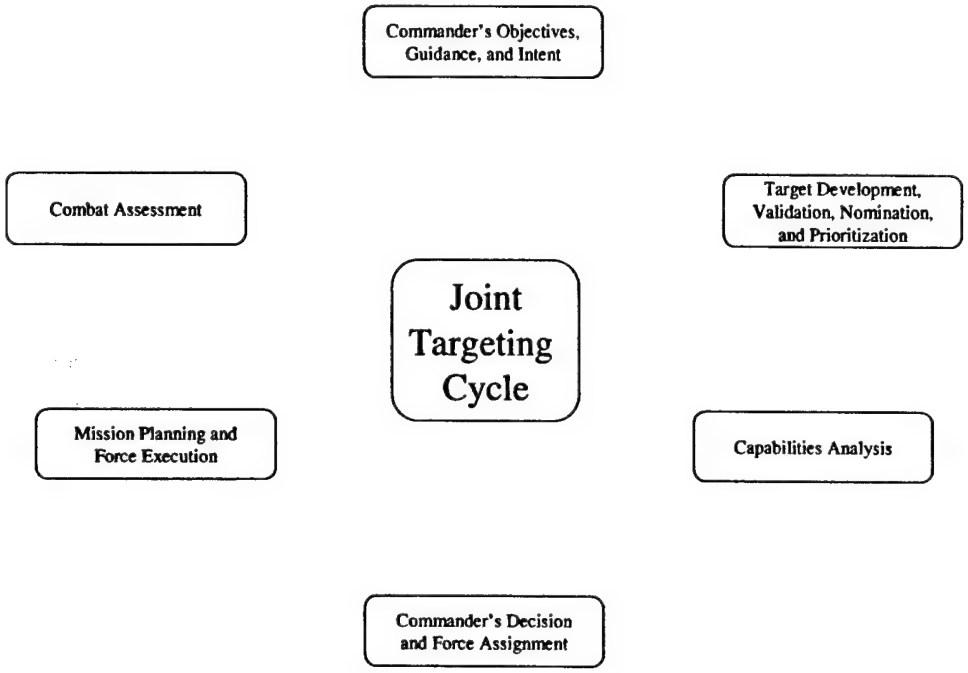

Figure 2-1: Joint Targeting Cycle

Once the targeting list has been compiled and resource assignments made, target-resources pairs are approved and orders disseminated under the *Commander's Decision and Force Assignment* phase. Upon completion of these detailed tasking orders, *Mission Planning and Force Execution* can take place. In this phase, final mission details are resolved and targets attacked.

Finally, *Combat Assessment* is the phase at which the effects of attacks from the previous phase are evaluated. This evaluation serves to determine if the target has been successfully destroyed and thus whether or not it needs to be included in the next round of target development. Battle damage assessment (BDA) is the method by which this evaluation is accomplished. A second function of this phase, munitions effectiveness assessment (MEA),

serves to determine the effectiveness of the assets and tactics applied to a target. Those determinations are then used as input for capabilities analysis.

In a situation where there are a large number of targets, the targeting cycle may be at different phases for different objects. In fact, it is this interaction between concurrent targeting cycles that motivates this research. All in all, the targeting cycle can be looked at as a useful framework. In this research we will focus upon phases two through six which are given in detail by the "Attack Mission Cycle" as defined by [4], shown in *Figure 2-2*.

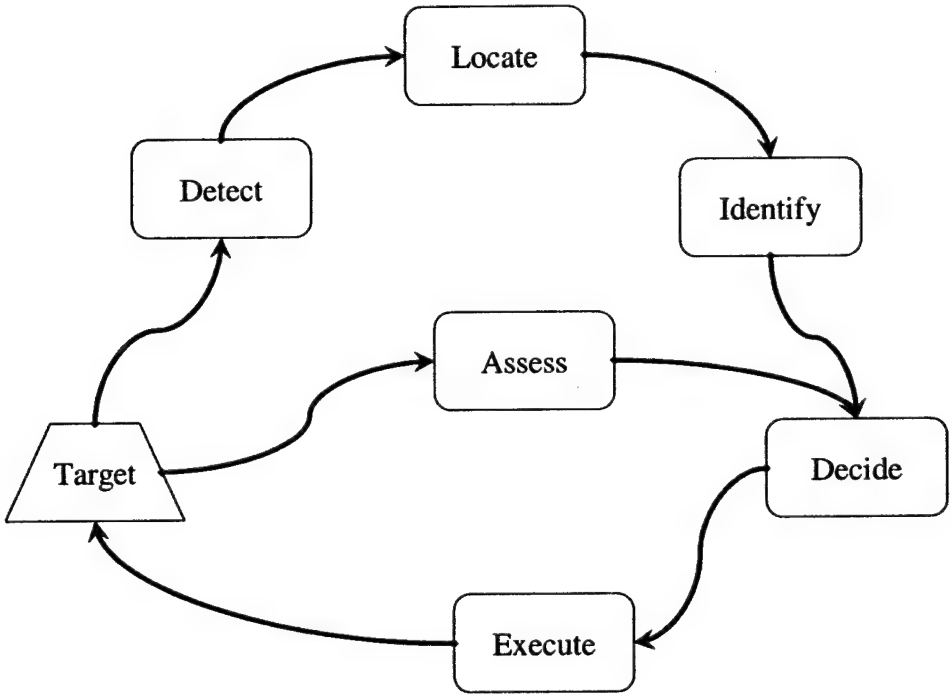

Figure 2-2: Air Force Attack Mission Cycle

2.1 Pre-Strike ISR

Detection, location, and identification of potential targets are the first tasks in the attack mission cycle. These segments of the cycle are accomplished by pre-strike intelligence, surveillance, and reconnaissance (ISR) missions, which can be accomplished by a wide variety of assets. For instance, satellites orbit the Earth constantly gathering information that can be used for ISR. In-theater operatives gather human intelligence. Somewhere in the middle of these two extremes is the use of aircraft to gather information. Manned aircraft missions performing ISR can be long, arduous, and dangerous.

In order to mitigate the risks to human life, unmanned aerial vehicles (UAV) are being used to complete such missions in place of manned aircraft such as the U-2. UAVs are aircraft that do not have a human in the cockpit controlling the flight of the aircraft. In the case of the RQ-1 Predator, human operators are at a ground station that can be far from the aircraft thus reducing the risk of human casualties. One operator controls the aircraft's flight while another monitors information gathering activities. An added advantage is the length of time Predator can stay in an area and collect information. Crews at the ground station can change out as often as necessary, but the Predator has an endurance time of 24 hours [3]. Such a long loiter time allows for almost constant surveillance of areas of interest.

While Predator has qualities that have led it to improved intelligence gathering operations, newer UAVs have better performance characteristics such as longer loiter times and less direct human control. The RQ-4A Global Hawk can not only takeoff, fly to the target area, perform ISR actions, return, and land autonomously, it also has an extended loiter time of almost 35 hours [2]. It also flies at high altitudes, up to 65,000 feet, and is thus less vulnerable to anti-aircraft systems.

These systems have expanded the role of airborne intelligence gathering assets in the modern battlefield. A commander can receive the latest intelligence about enemy actions and plan an appropriate response. Pre-strike intelligence gathering actions that support the commander can be divided into two phases: detection and location of objects in an area of interest and identification of contacts. The first two actions are grouped together because they can, and usually are, accomplished by the same platforms at the same time. For instance, a Global Hawk with Synthetic Aperture Radar can find and locate a large number of objects over a given region. In fact, a single Global Hawk can "image an area the size of Illinois (40,000 nautical square miles) in just 24 hours" [2].

2.1.1 Detection and Location in an Area of Interest

An area of interest can be defined as a geographical area in which there may be targets that need to be destroyed so that the commander's objectives can be accomplished. The size and shape of such an area can vary based upon terrain, geopolitical considerations, and other factors. A standard measurement is a grid square, normally defined to be 1 kilometer by 1 kilometer. This, however, is not a steadfast rule. In fact, areas of interest in the same battle space could be

of completely different size and shape. This makes it extremely important to have flexible assets that can perform well no matter how the areas of interest are formed.

Another essential characteristic of an area of interest is the potential for targets in the area. This can be measured as the expected number of targets in the area or as the probability mass function (PMF) over the possible number of targets in the area. In addition to information estimating the number of targets in an area it is also important to know the value of those targets. For example, intelligence sources might indicate there is a high concentration of armored personnel carriers in an area of interest. However, if there is no plan to move ground forces to the area of interest, gathering more information about the concentration of personnel carriers may be of little value. On the contrary, an area of interest that may contain a handful of surface-to-air missile (SAM) sites could be of high value because of the significant threat these SAM pose to our aircraft.

In order to "Detect" and "Locate" targets, as part of the Air Force attack cycle, *Figure 2-2* and "Target Development, Validation, Nomination, and Prioritization" in the targeting cycle, *Figure 2-1*, path planning must be done. During path planning, the route an aircraft will fly is determined. This is done after resources have been assigned to the mission so that platform specific attributes can be taken into account. ISR missions to areas of interest with SAM sites will be planned differently for a Global Hawk than for a Predator due largely to the Global Hawk's high cruising altitude. That is beyond the range for all but the most sophisticated of enemy surface-to-air defenses. Predators, on the other hand, have a maximum altitude of 25,000 feet [3] which is well within the range of a SAM. Consequently, different search patterns may need to be drawn up for an area of interest based upon the type of aircraft used. Once a search pattern is established, specific guidance would be given to human operators to ensure the best possible search is accomplished while limiting the risk to the aircraft.

2.1.2 Identification of Contacts

When an object has been discovered and its approximate location found, the next step is to identify its target type as shown represented by the "Identify" phase of *Figure 2-2*. Identification of contacts is important when strike actions have the potential for inducing collateral damage of noncombatants and non-military structures. Such situations are becoming more and more prevalent. In the past decade, cities such as Baghdad, Mogadishu, and Sarajevo

were focal points of large scale skirmishes. In such large cities, where civilian and military assets are in such close proximity, correct identification of a contact, to prevent civilian casualties, is both a military and a political issue.

As with contact identification in urban areas, it is important in areas where there are only military assets. Correct weaponeering is a driver behind this need for proper identification. Weaponeering considerations are important because of the advanced, specialized armaments used by today's military. For example, the AGM-65 Maverick missile is "a tactical, air-to-surface guided missile designed for close air support, interdiction and defense suppression mission" [1]. During Desert Storm, Maverick missiles were employed "mainly attacking armored targets" [1]. On the contrary, cluster munitions, such as the CBU-52B, are "best used against personnel or light-skinned vehicles" [5]. Thus, it is important to correctly identify a contact so that the appropriate munition can be used against the target. If a less effective munition is used, the odds of destroying the target are greatly reduced.

Conservation of munitions is the final driver for accurate identification of contacts. As shown above, misidentification could lead to over use of weapons. A lack of timely information could also lead to the over use of weapons, especially when dealing with mobile targets. While an object might be found in an area of interest that contains only military assets, strike resources may not be available to prosecute such a target right away. If the time between initial discovery and prosecution is even a few hours, a target may move from the original location. Sending a strike mission to destroy a target that has moved is a waste of valuable weapons. Also, it induces unnecessary risk of losing an aircraft. Thus, it is important to collect information on a contact until shortly before it is prosecuted. That way, resources will not be wasted and the maximum benefit can be attained.

2.2 Strike

When a target has been found in an area of interest and has been correctly identified, the decision whether to strike is made. This follows directly from the "Decide" and "Execute" phases of the Air Force Attack Cycle, *Figure 2-2*, and the "Mission Planning and Force Execution" phase of the Joint Targeting Cycle, *Figure 2-1*. If the decision to strike a target is made, numerous additional details must be worked out. In addition to weaponeering, decisions are made about path planning, timing, individual aircraft assignment, necessary supporting

aircraft, and other such issues. The strike action is the only point in the cycle when the target is directly influenced and thus proper execution is critical.

2.3 Post-Strike ISR/BDA

After a target has been attacked, the next step is to determine the target's functional state. Rarely would it be acceptable to consider a target destroyed without confirmation of weapons delivery or data indicating the target's destruction. Such information might come from the combat pilot who performed the mission. It might come from another aircraft with ISR equipment that can take pictures, detect radio frequencies, or produce thermal images. Information can even come from assets such as satellites or human operatives. No matter from where the information comes, it is invaluable to the targeting process, allowing the commander to decide whether the target needs to be struck again. This decision is not as straightforward as it might seem. Determining if a tank has been destroyed, either by seeing smoke and fire coming from the tank, a lack of radio signals, or a number of other indicators may be easy; determining if a building or runway has been sufficiently destroyed can be somewhat harder. During Desert Storm, prime targets were bridges. If two of the four spans of a bridge have been destroyed, is that bridge 50% damaged or 100% destroyed?

Not only must battle damage assessment (BDA) be gathered for use in determining target status, it is also important for future weaponeering decisions. While AGM-65 Maverick missiles may be the primary munition employed against armored targets, BDA could show that other munitions could be equally or more effective against certain armored targets. This would allow limited resources to be used in the best manner possible while still allowing the commander's objectives to be met.

Another motive for post-strike ISR is that some targets may be seriously damaged by a strike but can be rebuilt. Military airfields are one example of such a target. A runway may be rendered useless by a cluster bomb. With the right construction equipment and supplies, the damage can be repaired and military operations could resume at that airfield. Accordingly, it is important to keep updating information on *regenerative* targets so that appropriate measure can be taken to ensure that they remain inoperable.

2.4 Chapter Summary

Each individual phase of the Joint Targeting Cycle and the Air Force Attack Cycle are essential but it is important to remember that they are part of a cycle. Each phase must be accomplished so that further phases in the cycle may take place. Increasingly the assets that are used to complete each phase of these cycles are the same. UAVs, such as Predator, have been used as intelligence gathering assets and as seen in Operation Enduring Freedom in Afghanistan, they can be used to deliver weapons. Strike platforms such as the F-16 and F-15E are being fitted with sensors so that they can perform the ISR mission in addition to striking targets. Given the limited availability of aircraft and weapons, we must determine how to best allocate these resources to the detection, location, identification, and destruction of enemy targets thus ensuring that command objectives are met.

3 Modeling the Targeting Cycle

In this section, we address two broad aspects of the Joint Targeting Cycle, *Figure 2-1*, that have been well structured but not effectively modeled. The first is the allocation of limited resources to accomplish specified missions. The second is to find the best actions to take against an object of a given type based upon current information about the state of all objects of interest and the available resources. First, though, we explore the problem characteristics associated with the targeting cycle.

3.1 Targeting Cycle Problem: Characteristics and Model Assumptions

In order to model and solve the *targeting cycle problem*, it is important to understand the problem characteristics and to make modeling assumptions. We define the targeting cycle problem as the problem of optimally allocating aircraft, weapons, and sensors to find, identify, destroy, and confirm the destruction of enemy objects. We define the state of the system as the state of our resources and enemy objects. While we focus upon the targeting cycle problem, problems with the following characteristics can be addressed in a manner similar to our treatment of the targeting cycle problem.

Discreteness and Finiteness of System Elements: In dealing with the targeting cycle problem, we require that the state of each enemy object, be it a target, contact, or area of interest, be part of a discrete set. The state representation for an area of interest is the number of objects

in the area and thus is discrete. Because we are dealing with a real-world problem, we know that the number of objects in an area is finite. *Figure 3-1* shows the state representation for an area of interest which is simply the integers from 0 to a maximum number of possible objects in the area.

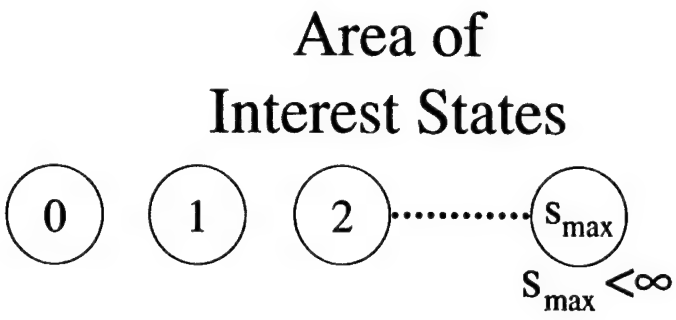

Figure 3-1: State Representation for an Area of Interest

A contact is a type of target, a collateral entity, or nothing at all, which are discrete states and make up a finite set in a real-world problem. *Figure 3-2* illustrates some possible states for a contact which include military assets as well as the possibility that a contact is not a target.

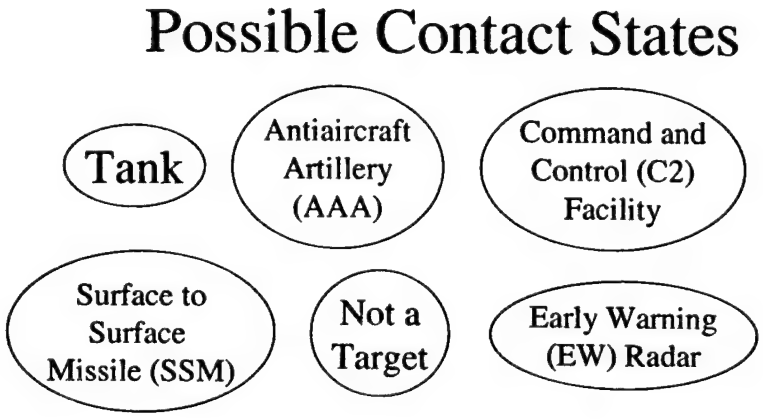

Figure 3-2: Possible State representation for a Contact

To fully model the state of a target, we would need an infinite number of states representing the percent damage to the target. Thus targets do not have an obvious discrete state space. In this work we assume this state space is discretized into a finite number of segments. A two state problem could be considered in which the target is either live or dead. This can be expanded to a three state case in which the target is alive, 50% damaged, or dead. *Figure 3-3* illustrates these two cases as well as a possible five state representation. Larger state representations would include more intermediate states between live and dead.

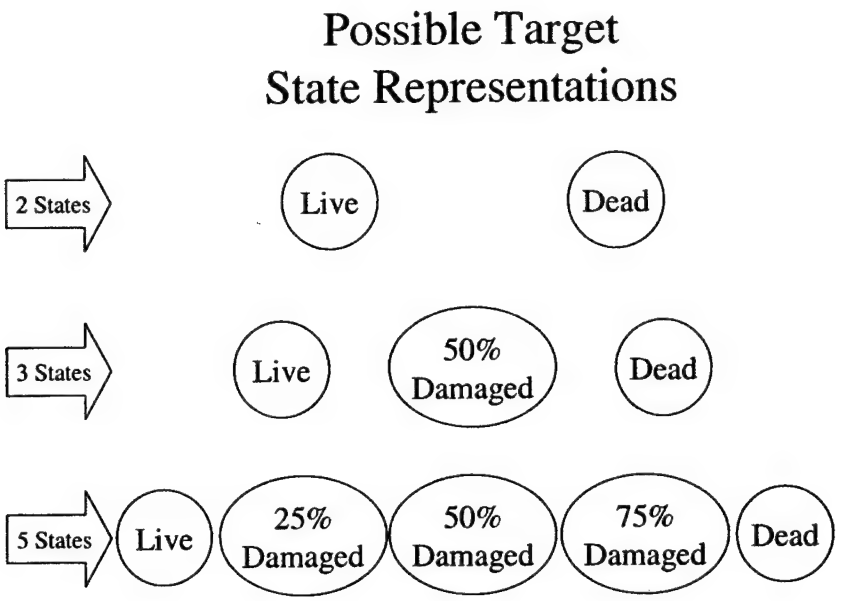

Figure 3-3: Possible State Spaces for a Target. Further states could be added to better model the true state of a target. The infinite state case would include 0% damaged, 100% damaged, and all values in-between.

Beyond the objects being acted upon, we assume that actions are planned and executed at discrete points in time with a finite planning horizon of T time periods. Finally, we assume that resources are used in finite discrete amounts.

Memoryless Stochastic System Evolution: Given that an object is in a given state s_t at time t, its state at time t+1 depends only upon s_t and the action applied at time t, ψ_t. In the targeting cycle problem this action could include the employment of aircraft along with weapons, sensors, or a combination of the two. This property is commonly called the Markovian property and mathematically is stated as:

$$P\left(s_{t+1} \mid s_0, s_1, \ldots, s_t, \psi_t\right) = P\left(s_{t+1} \mid s_t, \psi_t\right). \tag{3.1}$$

In addition to assuming that the state of each object we are acting upon evolves in this manner, we also assume that our resource levels evolve in this way. For instance, when an aircraft is sent out to strike a target, we know that there will be fewer available weapons for use in the next time step. It is also possible that an aircraft is shot down. Assuming the Markovian property is true for the targeting cycle problem, the number of aircraft available for use at the next time step depends only upon the number currently available and the actions performed in the ensuing time step.

Imperfect State Information: While we know the resources that are currently available, the true state of enemy objects is unknown. This stems from the fact that our ISR assets are not perfect observers of enemy object types or changes in their state due to our actions. For example, an ISR asset may misidentify a tank as an artillery piece, or any number of other similar systems. Imperfect information prevents us from making determinations based upon the true state of the system. Rather, we act upon our belief about the state of the system.

Independence: We assume an object's state at a given time is *independent* of the state of another object at any time. We also extend this assumption to the outcome of our actions applied to an object. If, for instance, a bomb was dropped on a target, its impact upon the target does not depend upon our past actions against that target or any other object. Such an assumption implies that we have a model of the weapon's effects that accurately describes the probabilities of kill for different types of targets. This does not mean, however, that munitions effectiveness assessment (MEA) does not need to be done via BDA as described in *Chapter 2*. Rather, MEA data is not immediately incorporated into the action model to maintain the independence assumption.

Linear Reward Structure: Due partly to the independence assumption, we assume that the reward for causing different objects to transition between states is additive, and thus the total value function is linear. For example, rewards are gained from identifying a contact or destroying a tank. If both of these actions take place, the total reward would simply be the reward for identifying the contact plus the reward for destroying the tank. This assumption and the independence assumption are somewhat questionable when there are enemy capabilities such as integrated air defense (IAD) because knocking out a central control facility may in turn reduce the effectiveness of subordinate air-defense sites.

Intelligent Adversary: During a battle, we are not the only one taking actions. Our enemy might be acting to limit or degrade our past, current, or future actions. As mentioned earlier, fixed targets, such as a runway, can be repaired; a contact might evade our sensors and move to another location. An undetected object moving between areas of interest is another action an intelligent adversary would take. However, this would violate our independence assumption because changes in the state of one area of interest would affect the state of another area of interest. Due to this violation we do not consider such movements in this work.

3.2 Current Modeling Methods

The targeting cycle problem as defined in *Section 3.1*, can be divided into two parts; the first is the allocation of resources and the second, the determination of specific actions to take against an object. The first part of the problem is addressed through *resource allocation methods* based upon mathematical programming. For the second, we consider *policy development methods* that can be solved using a dynamic programming framework. Finally, we consider a hybrid method in which the two parts of the problem are solved in an integrated formulation, as shown in *Figure 3-4*. Details are presented in the following three subsections.

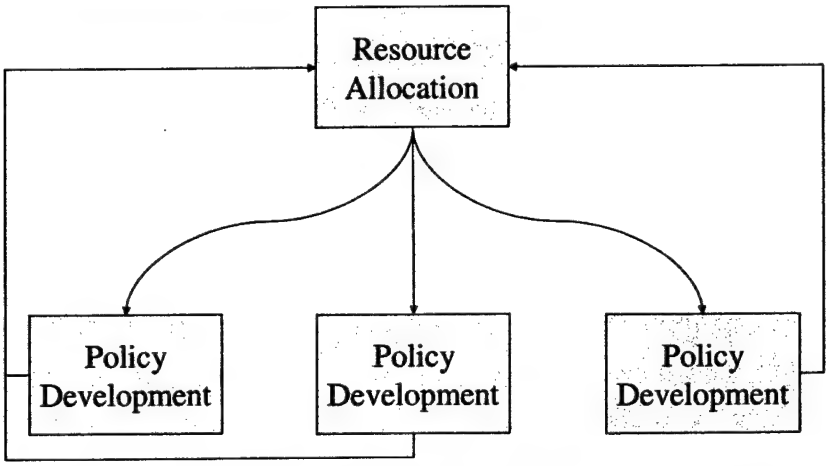

Figure 3-4: Hybrid Resource Allocation and Policy Development Method

3.2.1 Resource Allocation Methods

Allocation of limited resources using mathematical programming methods has long been used in the operations research (OR) community. When we consider the four primary characteristics of linear programming (LP) problems, as described by Hillier and Lieberman [23], we see that the targeting cycle problem fits into such a framework quite well. Proportionality, in both the objective function and subsequent constraints, is described as "the contribution of each activity to the value of the objective function...is proportional to the level of the activity..." and "...the contribution of each activity to the left-hand side of each functional constraint is proportional to the level of the activity..." [23]. Secondly, in order to cast our problem as an LP, we must have additivity, defined in that the objective function and constraints are "the sum of the individual contributions of the respective activities" [23]. Divisibility, the third characteristic of an LP, is when "Decision variables...are allowed to have any value,

including non-integer value..." [23] provided that they satisfy the problem constraints. Conceptually, the targeting cycle fulfills this requirement but in practice, we cannot send partial aircraft or weapons. Rather, when this model is solved in an operational setting, it is formulated as a mixed integer program (MIP) or an integer program (IP). Solutions from these types of problems can be directly executed by forces in the field.

Lastly, linear programs are assumed to follow the certainty principle. That is, the coefficients of the decision variables are known constants. This, however, is not the case in the targeting cycle problem. We can only estimate the true reward we will receive and the resources that will be used by taking a sequence of actions. Even so, this does not mean that the problem cannot be solved as an LP. It has been noted by many that "...the certainty assumption is seldom satisfied precisely" [23] in real applications. The field of sensitivity analysis deals with questions about the impact of changes in LP data.

To formulate the targeting cycle problem as an LP we define the following notation. Each object is indexed by $i \in \mathrm{I}$. Let $\mathrm{O_i}$ be the set of possible contingency plans for object i. *Contingency plans* are fully developed in *Chapter 4* but the basic notion is that they are a mapping from the current state of an object to a sequence of actions contingent upon observations received about the object's state. Let $\mathcal{U}_{joi}$ be the resources of type $j \in \mathrm{J}$ used by contingency plan $o \in \mathrm{O_i}$ against object i and $\mathcal{Y}_j$ be the resources of type j available for use. Finally, define x_{oi} as the proportion of contingency plan o to use against object i and $\mathcal{R}_{oi}$ as the associated reward. Our formulation thus becomes:

$$\max_{x} \sum_{i \in I} \sum_{o \in O_i} \mathrm{E}[\mathcal{R}_{oi}] x_{oi} \tag{3.2}$$

$$\text{s.t.} \quad \sum_{i \in I} \sum_{o \in O_i} \mathrm{E}[\mathcal{U}_{joi}] x_{oi} \le \mathcal{Y}_j \quad \forall j \in J \tag{3.3}$$

$$\sum_{o \in O_i} x_{oi} = 1 \quad \forall i \in I \tag{3.4}$$

$$x_{oi} \ge 0 \quad \forall o \in O_i, \; i \in I. \tag{3.5}$$

The objective function (3.2) seeks to maximize the expected reward by applying contingency plans from the set $\mathrm{O_i}$ to object i. Resource usage is constrained in expectation (3.3) and the requirement to fully act against an object is enforced (3.4). The decision variables can be thought of as the proportion of contingency plan o to use against object i thus, these values must

be between 0 and 1, as specified by constraint (3.5). In *Chapter 4, Section 4.4.4* we discuss a mixed integer programming (MIP) formulation of the targeting cycle problem which yields an executable solution.

By using the expected reward and resources usage, we have an LP formulation that satisfies all four of the primary characteristics. However, the sets O_i for all i have yet to be defined. For a given object i, $|O_i|$ is the number of columns in the LP corresponding to i and thus its size is of extreme importance. While computing advances in hardware and software have greatly increased the size of linear programs that can be solved, a large number of decision variables can greatly increase solution times especially when an integer solution is needed. Models with long solution times are ineffective for real world situations in which we need to plan and replan in short periods of time.

Define Ψ_i as the set of allowable actions for object i and Θ_i as the set of possible observations that could be received in a single period through ISR. For a horizon of one, it is clear that the number of contingency plans is $|\Psi_i|$. However, for a horizon of T larger than one, the number of contingency plans, C_T is

$$C_T = |\Psi_i| C_{T-1}^{|\Theta_i|}. \tag{3.6}$$

The reasoning behind this equation is that for a horizon of 1, we simply consider taking each action. The resulting observations are not considered because we cannot act upon them. For longer time steps, we must chose an action for every observation at each level. We can use the one-step contingency plans to build the two-step contingency plans, the two-step contingency plans to build the three-step and so on. *Figure 3-5* shows the one- and two-step contingency plans for a problem with two actions and two observations.

Even if Ψ_i and Θ_i are relatively small, a long horizon can make the number of possible contingency plans enormous. As observed by Yost [37], actions that do not provide relevant information have only one follow-on possibility and thus C_T is somewhat reduced. However, for objects on which only ISR actions are to be performed, the number of contingency plans quickly makes the problem become intractable as Ψ_i, Θ_i, and/or the horizon increase. To get an idea of how large C_T can become, even when there are non-ISR actions, Yost provides the example shown in *Table 3-1* that includes 5 actions, 2 of which conduct ISR, and 2 possible observations.

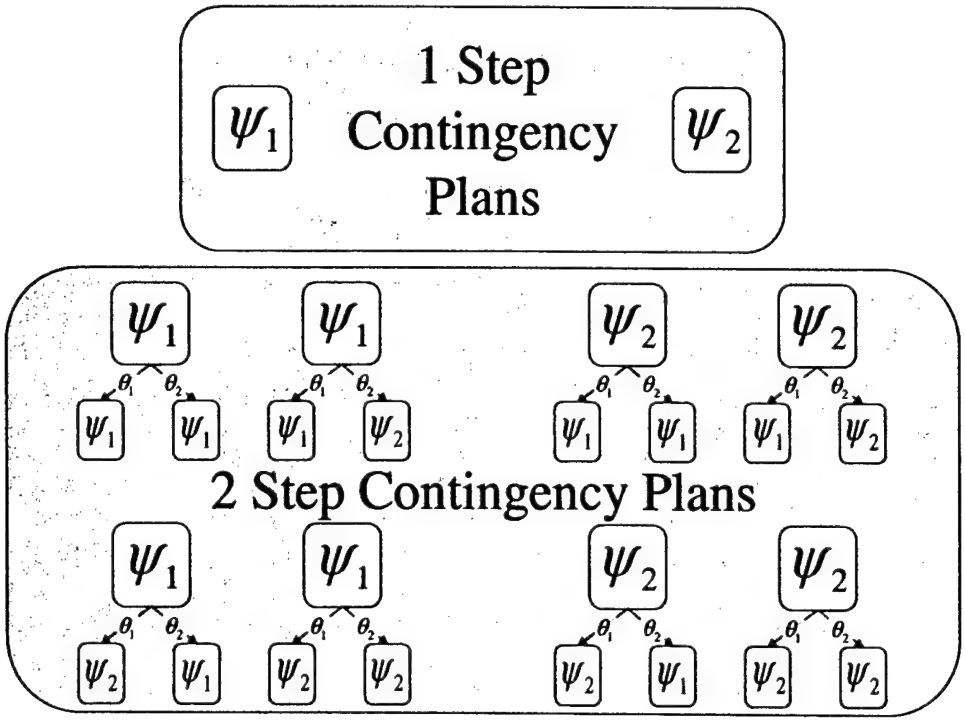

Figure 3-5: One- and two-step contingency plans for a problem with two actions and two observations. Note the combinatorial growth in the number of contingency plans.

Planning Horizon	1	2	3	4	5
Potential Contingency Plans	5	65	8645	1.49 E8	4.47 E16

Table 3-1: Potential contingency plans for object with 5 actions, 2 of which conduct ISR, and 2 possible observations. Note the reduction from the possible number of contingency plans when all actions perform ISR.

While this may seem large, consider *Table 3-2* for a contact that has 5 actions, all of which conduct ISR with 4 possible observations.

Planning Horizon	1	2	3	4	5
Potential Contingency Plans	5	3125	4.77 E14	2.58 E59	2.23 E238

Table 3-2: Potential contingency plans for a contact with 5 actions, all of which conduct ISR with 4 possible observations

Linear programs with this many variables cannot be solved. A realistic problem could include twenty or more objects to act upon with dozens of actions and would have a planning horizon of 8 to 12 time steps depending upon step length.

A key observation that enables us to use the LP formulation is that many possible contingency plans are of little value and only a small subset need to be included in the LP. If we consider only those contingency plans, and their associated variables, which are selected in the final LP solution, the LP can easily be solved. There are heuristic methods by which contingency plans can be generated and used in the LP. For example *Figure 3-6* illustrates a

common target contingency plan to strike first, perform an ISR mission, and then strike again if the observation indicates that the target survived the initial attack.

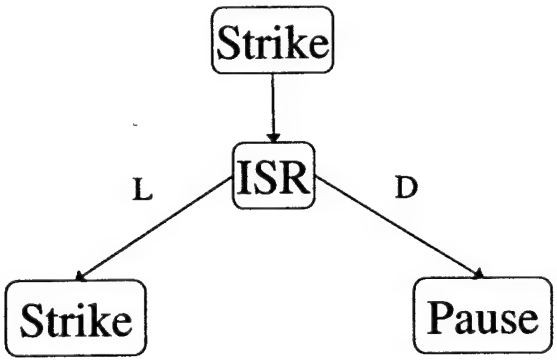

Figure 3-6: Sample strike, ISR, strike contingency plan for a target. L represents a live observation and D a dead observation.

A heuristic method, however, provides no guarantee that the columns considered include all of the contingency plans that are part of the optimal solution. This lends itself to the idea of column generation which Gilmore and Gomory define as "...'pricing out' or looking for a new column or activity that will improve the solution..." [20]. In order to find an activity, and thus a new column, that improves the solution, "...we simply create a useful column by solving an auxiliary problem" [20]. By using a sub-problem to search through the possible contingency plans and then only considering those that "price out" favorably, the computational effort can be greatly reduced. Common sub-problems that are used in column generation are shortest path and knapsack problems [8]. Neither of these sub-problems addresses the stochastic and partial observable nature of the targeting cycle problem.

Stochastic programming, as described by Birge and Louveaux [9], is another allocation method that addresses the problem of sequential decision-making. The general stochastic programming problem is multi-stage stochastic programming with recourse. This formulation models the system of interest in stages such that decisions are made and then stochastic events occur. The realizations of these stochastic events influence the value of subsequent decisions as well as the feasibility of decisions at later stages. These are characteristics of the targeting cycle problem. The decisions to be made are the assignment of aircraft, weapons, and sensors to perform certain missions against specific objects. Stochastic events in the targeting cycle problem include: the transitioning of objects from one state to another, the loss of aircraft and sensors because of attrition, and the observations from sensor platforms. A key assumption of

stochastic programming is that the realizations of stochastic events are independent of previous actions. This assumption is not met in the targeting cycle problem because the likelihood of an object being in a given state is dependent upon ISR or strike actions from previous periods. In fact, the reason for modeling the targeting cycle is to determine which actions provide the best outcomes. Stochastic programming is therefore not an option for our model.

3.2.2 Policy Development Methods

The resource allocation methods described in *Section 3.2.1* fail to adequately model the targeting cycle problem in two main areas: fundamentally, they do not address the probabilistic transitions and observations and practically, dealing with the large number of contingency plans is not feasible with the current LP solver technology. Rather than allocating resources, policy development methods find the best policy for a given object. Policies will be fully explained in *Chapter 4* but the basic idea is a mapping from states to actions for all states and time steps. A logical starting point for policy development methods is Dynamic Programming (DP). DP is used to solve models that have two main features. First there must be a core discrete time dynamic system and second, the costs must be additive over time [7]. These restrictions may be relaxed or further restrictions can be placed upon system dynamics, the cost structure, or other parameters, but these two properties form the basis of dynamic programming. For each state of the system at a given time, the immediate cost plus discounted future costs is minimized or similarly, discounted future value maximized. Such a method is attractive for the targeting cycle problem because the two basic assumptions are satisfied.

We can make use of additional problem characteristics to further refine the solution method. The Markovian nature of the state evolution allows the targeting cycle problem to be modeled as a *Markov Decision Process* (MDP). **A MDP is a decision process, with an underlying Markov chain, for choosing the optimal actions for a set of states considering both immediate and future rewards**. A MPD has four main elements:

1. **A set of states,**
2. **A set of actions for each state,**
3. **Action-dependent Markovian state transition probabilities,**
4. **A reward structure indicating immediate rewards as well as terminal rewards for each state.**

In solving an MDP, an optimal policy, described as a "...specification of the decisions for the respective states..." [23], is generated such that it maximizes the expected reward, based upon the system's evolution over time. This optimal policy is then stored in a table that can be referenced as the system evolves.

MDPs are typically solved using dynamic programming but can also be solved using linear programming [26]. This shows that MDPs can be solved in polynomial time, per the proof that linear programming is contained in P [8], and thus may be useful in solving realistic targeting cycle problem scenarios. Meuleau et al. [30] used MDPs to solve a "military air campaign planning problem" in which "tasks correspond to targets" and "there are global constraints on the total number of weapons available...(and) the number of available aircraft." Meuleau also assumes that "actions have inherently stochastic outcomes and the problem is fully observable." The characteristics of the military air campaign planning problem are the same as the targeting cycle problem except that we do not have full knowledge of the state of the objects being acted upon. MDPs do not capture enough fidelity of the problem because the true state of the objects being acted upon is partially observable due to imperfect sensors, enemy actions, and other sources of "friction" [14].

Partial observability in MDPs has been considered and studied for over thirty years. Originally proposed by Drake [16] and formalized by Sondik [34], the partially observable Markov decision process (POMDP) is an extension of the MDP model. **A POMDP is a decision process, with an underlying Markov chain, for choosing the optimal actions for a set of states considering both immediate and future rewards, in which the true state is not known but rather is partially observable.** A POMDP has six main elements:

1. **A set of states,**
2. **A set of actions for each state,**
3. **Action-dependent Markovian state transition probabilities,**
4. **A set of observations,**
5. **Action-dependent observation probabilities mapping observations to states,**
6. **A reward structure indicating immediate rewards as well as terminal rewards for each state.**

These elements are more precisely defined and related to the targeting cycle problem in *Section 3.3.1.*

Sondik states that partial observability may arise "...if the observer is removed from the process in some sense and must receive his information over an imperfect communication channel" [34]. The true state of the battlefield is partially observable because commanders rely upon intelligence analysts who look at information from ISR platforms, talk with pilots who perform the missions, gather human intelligence, and otherwise analyze the state of enemy objects. Using this information along with other factors such as personal experience, previously reported information, and higher level command objectives, the commander makes a decision as to what actions to take. Not knowing the true state of enemy forces, such a decision is based upon the commanders "belief" about the state of the system. *Belief states* become more important when solving a POMDP and interpreting the resulting policy.

Sondik describes a machine inspection and replacement problem in which a tire production machine can be in two states, "Good" or "Fail" with three possible user actions, "Operate," "Inspect," and "Replace." If the replace action is taken, the machine is assured to move to the good state after which a new tire is produced. With a known probability, P_F, the machine will transition from the good to fail state if operated or a tire is inspected, see *Table 3-3*. If an inspection occurs, information is gathered by looking at a tire produced by the machine. The machine produces a bad tire when in the good state and produces a bad tire when in the fail state with known probabilities. Each action has associated costs dependent upon the state of the machine. The objective is to minimize the expected total cost over a specific, possibly infinite, horizon by selecting which action to take based upon the current belief that the machine is good. *Table 3-3* shows the transition and observation probabilities for Sondik's machine inspection and replacement problem. Note that the observation probabilities for the "Operate" and "Replace" actions are uniform over the states. This occurs because the POMDP framework requires an observation to be received at every time step. In this case, however, we do not obtain useful information from the "Operate" and "Replace" actions. *Figure 3-7* shows the states and possible observations in Sondik's machine inspection and replacement problem.

Action	P(transition from good to fail)	P(transition from fail to good)	P(observe good tire given good machine)	P(observe poor tire given good machine)	P(observe good tire given failed machine)	P(observe poor tire given failed machine)
Operate	P_F	0	0.5	0.5	0.5	0.5
Inspect	P_F	0	P_{GG}	P_{PG}	P_{GF}	P_{PF}
Replace	0	1	0.5	0.5	0.5	0.5

Table 3-3: Transition and Observation Probabilities for Sondik's Machine Inspection and Replacement POMDP. P_{GG}, P_{PG}, P_{GF}, and P_{FF} are problem characteristics based upon the probabilities listed in the heading.

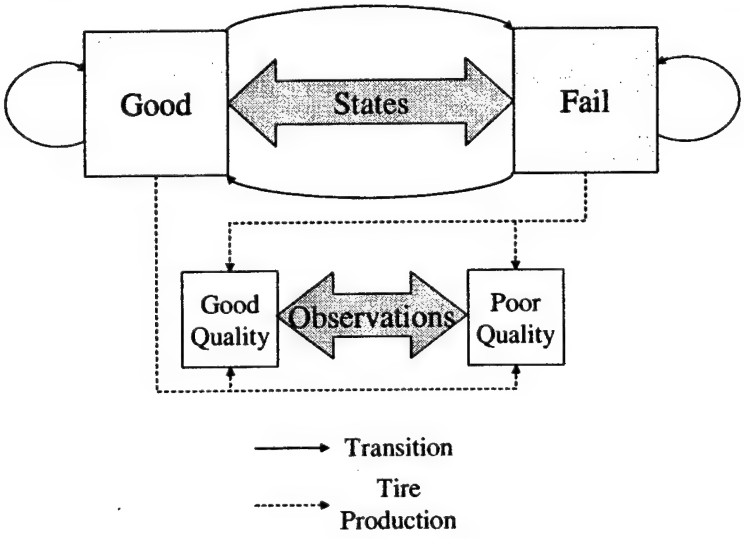

Figure 3-7: Sondik Machine Inspection and Replacement Problem: States and Observations

One important note is that the number of possible observations is not tied to the size of the state space, as is the case in the machine Inspection and Replacement Problem which has 2 states and 2 observations. There can be as many or as few observations as the system dictates as long as a probability mapping between states and observations can be made.

Castañon [12] uses POMDPs to consider a problem similar to contact identification in which there are a large number of objects of unknown type. Sensors are used to gather information about the objects and then classify them as one type or another. In dealing with this problem, Castañon decomposes the problem hierarchically. At the lower level, for each object, the best policy is determined by solving a POMDP given a cost for resources. The upper level maintains the expected resource usage constraints and determines the appropriate resource costs. To determine the appropriate resource costs, λ, the method runs "...a line search on λ to determine a value of λ such that the surplus is nearly zero" [12]. Updated resource costs are passed to the sub-problem POMDPs, which then generate a new policy based upon these costs.

The method iterates until the optimal set of strategies has been found. Such an approach provides a valuable example of sensor assignment for contact identification using POMDPs but the problems of target prosecution and object detection are not included. They are, however, mentioned as potential further research.

Castañon's approach, and more specifically the cost estimation phase, demonstrates a key component of policy development methods. In solving POMDPs it is assumed that there is a known cost structure. This is not the case in the targeting cycle problem. Rather, we have resource constraints to satisfy. Assigning a value to resources based upon these constraints is needed in order to use POMDPs to generate policies for use in the targeting cycle problem.

3.2.3 Hybrid Method

Allocation methods adequately address the resource constraints and the basic stochastic nature of the targeting cycle problem, but the large number of variables creates tractability issues. Some policy development methods address the stochastic and even the partial observable characteristics of the problem. However, resource costs, rather than resource constraints, are used, limiting their usefulness in the context of resource allocation problems. Each area can deal with portions of the targeting cycle problem. Used together, they provide a complete solution methodology for the targeting cycle problem.

Yost [37] uses linear programming to maximize the objective function, equation (3.2), and deal with resource constraints, equation (3.3), while generating columns associated with optimal policies for target-type POMDPs. While providing the LP with improving *contingency plans*, the POMDPs receive action costs based upon the duals from the resource constraints, (3.3), in the LP. The algorithm terminates when the solution is sufficiently close, ε, to optimal. *Figure 3-8* illustrates Yost's decomposition, which is initialized with columns from a heuristic policy generator, and is formulated using a master LP with objective function equation (3.2) and constraints, equations (3.3) to (3.5). Improving contingency plans are generated using POMDPs, one for every type of target in the sensor-shooter problem. The iterative algorithm terminates when the percent difference between the upper and lower bounds is less than ε. The upper bound is based upon the POMDP solutions and the lower bound is the LP objective function value.

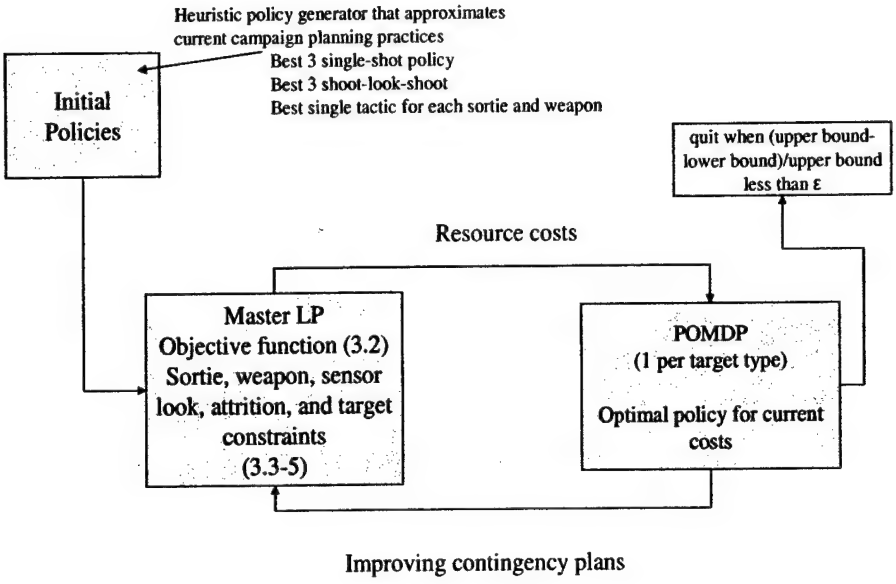

Figure 3-8: Yost hybrid decomposition with master LP and POMDP sub-problems.

Key to this method and its ability to solve the sensor-shooter problem, a simplified version of the targeting cycle problem with only targets, is the fact that POMDPs are run for each target type, such as tanks, command and control (C2) facilities, bunkers, etc. This can occur because of Yost's assumption that targets of the same type will necessarily have similar characteristics, namely value and Ψ_i. With this assumption, it is possible to solve one POMDP and use the policy it generates to build contingency plans for each individual target; this increases the size of problems the algorithm can handle. *Figure 3-9* illustrates the framework for a target contingency plan that Yost's algorithm would build based upon a POMDP policy. Each node has an associated optimal action and subsequent nodes for each possible observation. Contingency plans are formally defined and further developed in *Chapter 4, Section 4.5.3.*

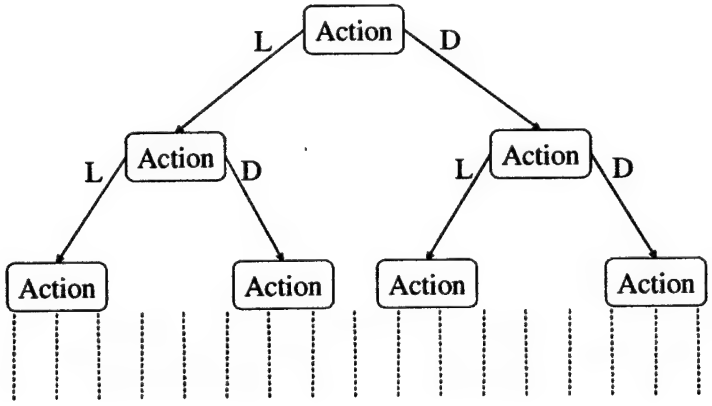

Figure 3-9: Sample Contingency Plan for Target with two observations, Live or Dead

Two main parts of the targeting cycle problem are missing from this formulation: *object discovery* and *contact identification*. Yost only deals with objects once they are found and identified. Because our resources can be used for both ISR and strike, the allocation of resources for object discovery, contact identification, strike, and BDA cannot be decoupled and thus must be solved together.

3.3 Completing the Cycle

A Yost-like hybrid approach seems to be appropriate to model and solve the targeting cycle problem. That approach needs to be enhanced and extended in order to adequately model the complete targeting cycle problem. Both *object discovery* in an area of interest and *contact identification* can be represented by additional POMDP sub-problems to be used in conjunction with the target sub-problems used by Yost. In order to do this we will first go into further depth concerning the POMDP model and its solution techniques.

3.3.1 POMDP Model

Before describing POMDPs and their solution methods, we define some notation. Define the set S to be the states of object i and Ψ_i as the set of allowable actions for object i. We define $\mathcal{E}_{ss'}^{\psi}$ to be the effects of action ψ, the probability that object i transitions from state s to state s' when action ψ is taken. Similarly for observations, $\mathcal{O}_s^{\psi\theta}$ is the probability of observing θ given that object i is in state s and action ψ produced the observation. We define $\pi(s)$ as the probability that object i is in state s. Finally, let r_{sk}^{ψ} be the reward for taking action ψ when object i is in state s at *epoch k*. While S, $\mathcal{E}_{ss'}^{\psi}$, $\mathcal{O}_s^{\psi\theta}$, $\pi(s)$, and r_{sk}^{ψ} are all specific for object i, we suppress the i to yield more concise notation for clarity of exposition.

The targeting cycle problem is a time dynamic system in which we must increment time in both the master LP and the sub-problem POMDPS. In describing POMDPs and their interaction with the LP allocation problem, we will be discussing *time steps* for the LP and *epochs* for the POMDPs. These are both time indices over the planning horizon T. POMDPs are solved via dynamic programming thus epochs begin at the end of the planning horizon and increase as we move closer to the current time. Epochs can be thought of as the number of steps

left in which actions can be taken. *Figure 3-10* shows the relationship between time steps and epochs.

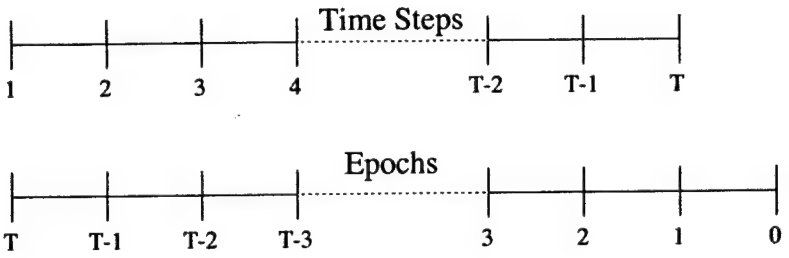

Figure 3-10: Comparison of Time Steps and Epochs

As stated before, POMDPs, and the first solution technique, were formalized by Sondik. In his dissertation a few key points were proven that provide the foundation for POMDP solutions. The first of these is the use of *belief states*, a probability distribution over the possible states. Solutions to Markov decision processes map a state to an action. If the true state of the system, as defined in *Section 3.1*, is not known, there is no way that such a mapping can be accomplished. We therefore transform the problem into one in which we know exactly what state we are in. That state will not be the state of the system but rather an *information state* for epoch k, I_k, that summarizes the complete history of the system. Such an information state includes the initial information about the system and all subsequent actions and observations. For the targeting cycle problem, we know the initial state of a target, presumably live, and then maintain a list of all actions, strike or BDA, that are performed on that target along with the observations these actions return. This representation leads to a huge state space that does not lend itself to finding the optimal set of actions easily, if at all.

Instead of using the information state, we characterize a *sufficient statistic*. Sufficient statistics are quantities that "...summarize all the essential content of I_k as far as control is concerned" [7]. Mathematically, the optimal action for an information state, $\psi_k^*(I_k)$, must equal the optimal action for the sufficient statistic π_k, $\psi_k^*(\pi_k)$, for π_k to be a valid sufficient statistic. Such a summary could reduce the size of the state space and allow for a mapping from state to action as was possible in the MDP. In the case of the POMDP, Sondik [34] proves that a *belief state* is a sufficient statistic. Although the information state space was large, it had a finite size. Once the transformation to probabilities (i.e., the belief state sufficient statistic) is made, the size of the state space is infinite. It seems that such a transformation has only compounded our

problem. This is not the case. For belief states that are relatively close, optimal actions might be the same. Thus it is possible to partition the belief space into portions over which single actions dominate. *Figure 3-11* illustrates this partitioning for three actions.

Probability theory provides us with the tools to manipulate belief states in this belief state space. Bayes' Rule allows us to update $\pi(s)$ to $\pi'(s)$ based upon taking action ψ and receiving observation θ as follows:

$$\pi'(s) = \frac{\sum_{s' \in S} \pi(s') \mathcal{E}^{\psi}_{s's} \mathcal{O}^{\psi}_{s\theta}}{\sum_{i,j \in S} \pi(i) \mathcal{E}^{\psi}_{ij} \mathcal{O}^{\psi}_{j\theta}}. \tag{3.7}$$

This equation can be simplified if the specified action does not affect the state of the object, in which case $\mathcal{E}^{\psi}_{ss'} = 1$ if s=s' and 0 otherwise, or it does not provide a meaningful observation, in which case $\mathcal{O}^{\psi\theta}_{s}$ is uniform over all states.

After the belief state, the next important property of POMDPs is the representation of a policy. As stated before, the transformation from an information state to a belief state made the size of the state space infinite. Thus, it is no longer feasible to store the state to action mapping in tables as is the case in finite-state MDPs. To define the policy and the associated value function, another representation is needed. We define r_{s0} as the *terminal reward* for state s and thus it is the same for all ψ. The terminal reward is defined as the reward received for being in state s at the end of the planning horizon.

Consider a problem in which there is only one time step for an action to occur. In such a case, maximizing the expected reward will be based solely upon the immediate reward for the chosen action and the terminal rewards. Therefore, the problem of finding the best action to take while in belief state π, the $\pi(s)$ vector, is simply a maximum over the actions in Ψ_i, of the immediate rewards plus the expected future rewards. For the one-step problem, that can be expressed in the dynamic programming form as:

$$\max_{\psi\in\Psi_i}\left\{\sum_{s\in S}\pi(s)r_{s1}^{\psi}+\sum_{s',s''\in S,\,\theta\in\Theta_i}\pi(s')\mathcal{E}_{s's''}^{\psi}\mathcal{O}_{s''}^{\psi\theta}r_{s''0}\right\}. \tag{3.8}$$

From this equation we can see that actions will dominate over a portion of the belief state. Thus we can represent the optimal policy as the partitions of the belief space over which each action dominates as shown in *Figure 3-11*.

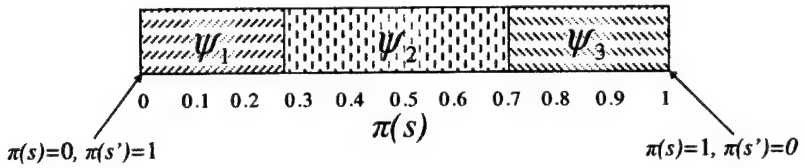

Figure 3-11: Partitioning of Belief Space due to Action Dominance

It is also important to note that the inner portion of the maximization is the value function for the POMDP. Via induction and algebraic manipulation, Sondik was able to prove that the value function for a finite-horizon POMDP is piecewise linear over the belief space. Due to the maximization, the value function is also convex. The argument stems from the fact that the zero-step value function is a linear function connecting each of the end states at r_{s0} for all $s\in S$ and, by definition, is convex. As shown in equation (3.8), the one-step value function will be a maximization over linear functions, thus is piecewise linear and convex. As the horizon of the problem increases, the number of linear functions that are being maximized over may increase but this does not change the property that the value function is piecewise linear and convex. The linear segments that make up the value function are called *alpha vectors* and are the basis for representing the optimal policies and value functions generated by a POMDP solution algorithm. Each alpha vector has value at the extreme points of the belief space and a corresponding action.

To better understand a POMDP value function, a geometric interpretation is useful. In fact, a geometric intuition about the POMDP value function will lead to a better understanding of the solution algorithms that will be presented in *Section 3.3.2*. We need further notation to develop this geometric intuition. We define $V_k(\pi)$ as the k^{th} epoch value function for the belief state π. Alpha vectors will be denoted as $\alpha^{\psi}(k)$ where ψ is the action associated with the alpha vector and k is the epoch. In addition, if a specific end-point value of an alpha vector at state s is needed, it is referred to as $\alpha^{\psi}(k,s)$. We index alpha vectors in epoch k by $u\in U(k)$ thus a

specific alpha vector from epoch k is denoted as $\alpha_u^w(k)$ or an end-state value for a specific alpha vector from epoch k, $\alpha_u^w(k,s)$.

Geometric representations of POMDP value functions are simple for two-state problems and, with standard plotting software, for three-state problems. However, for problems with four or more states, the geometric representation becomes a mental exercise. It would seem as though a two-state problem would require a graph in three dimensions, one for each state and one for the functional value. However, we are dealing with a probability space so this is not necessary. Such a two-state problem can be represented in two dimensions because the probability of being in one state is simply the complement, $1-\pi(s)$, of being in the other state. Therefore, our belief state in a two state problem reduces to a scalar representing the probability that the system is in a given state. An example of such a value function is shown in *Figure 3-12.*

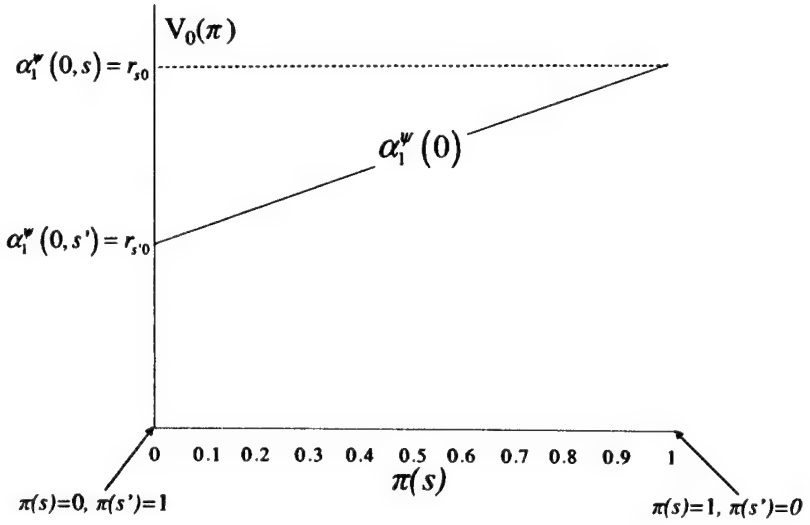

Figure 3-12: 0-Epoch Value Function

This value function represents the final epoch in a POMDP solution. For that reason, there is only one alpha vector and it has no associated action. It is from this single alpha vector that value functions for further epochs will be built. This procedure is discussed in *Section 3.3.2.1.* A k-epoch value function might take the form shown in *Figure 3-13.*

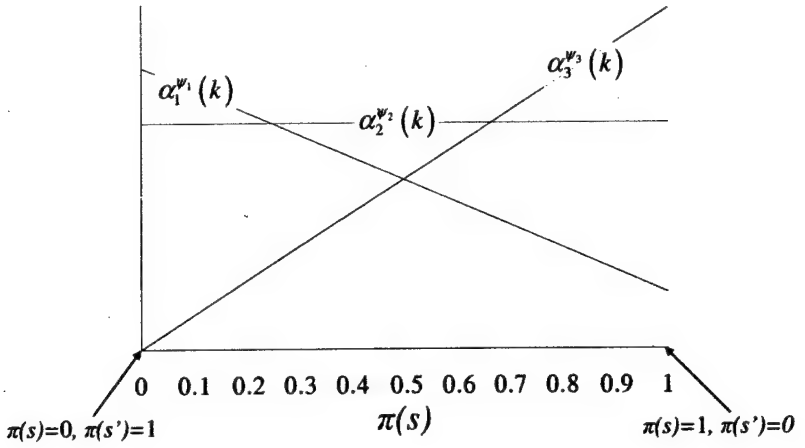

Figure 3-13: Alpha vectors making up $V_k(\pi)$

A noteworthy characteristic of this value function is that the belief space is divided into three portions. In the first section of the belief space, the action associated with alpha vector 1 dominates while the action associated with alpha vector 2 dominates in the second section and the action associated with alpha vector 3 dominates in the last section. Thus the actual $V_k(\pi)$ is the upper envelope of this set as shown in *Figure 3-14*.

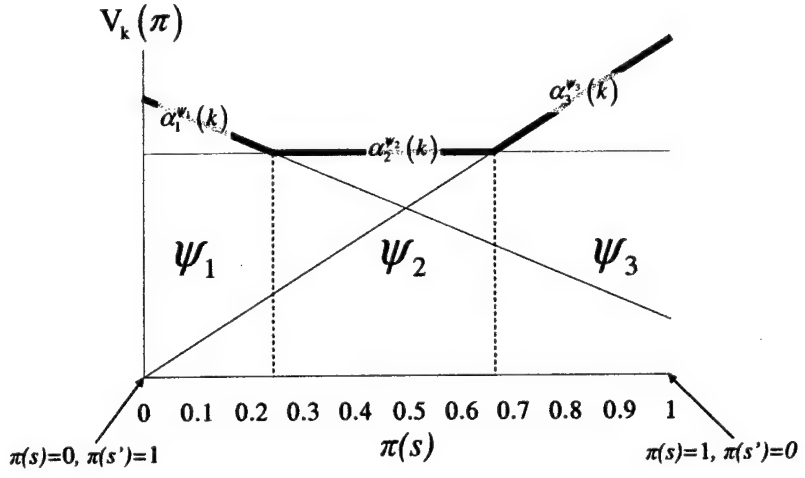

Figure 3-14: Value function for epoch k seen as upper envelope of alpha vectors

Using the above graph, it is possible, for any belief state, to determine the best action based upon current rewards and expected future rewards. For epoch k, developing the policy, and thus the value function, that make up this graph based upon Ψ_i and the value function for epoch k-1, $V_{k-1}(\pi)$, seems as simple as running a line search over the belief space and finding the points at which the optimal action changes. This, however, depends on some fairly restrictive structural properties of the optimal solution. Therefore, it seems best to pursue

algorithms for solving targeting cycle problem POMDPs based upon the general POMDP framework and not upon problem specific characteristics.

3.3.2 POMDP Solution Algorithms

Solution algorithms for POMDPs have been researched since Sondik proposed the first general POMDP algorithm, the "One-Pass Algorithm" [34]. Since then, a wide variety of researchers have proposed solution algorithms. Though initially an outgrowth of the operations research MDP literature, POMDPs quickly fell out of the OR literature. The Artificial Intelligence (AI) community recognized the potential of POMDPs and became the driving force behind recent algorithmic advances.

As with an MDP, a POMDP can be solved for a finite or an infinite horizon. Solving an infinite horizon problem is the same as solving a finite horizon problem for a sufficiently long horizon so that the optimal policy does not change. Such a solution is what the artificial intelligence community hoped to have because there was no set horizon over which the agent will act. Rather, it will continue to act until a goal is met. This is definitely not the case in the targeting cycle problem. In fact, we are looking for a dynamic solution that incorporates and deals with the inherently uncertain nature of the modern battlefield. We will therefore limit our discussion of POMDP algorithms to a finite horizon. Many of the algorithms presented in *Sections 3.3.2.1* and *3.3.2.2* can be implemented with a discount factor to solve infinite horizon problems but the reader is referred to Cassandra [10] and Cassandra, Littman, and Zhang [11] for further explanation of the infinite-horizon versions of these algorithms.

All finite-horizon POMDP solution algorithms follow a general framework as shown in *Figure 3-15*. They begin by building the 0-epoch value function using the terminal values, r_{s0}. A dynamic programming update is performed to find the next epoch's value function. These updates continue until there is a policy for each epoch in the planning horizon. The method by which the dynamic programming update is done is what distinguishes each algorithm.

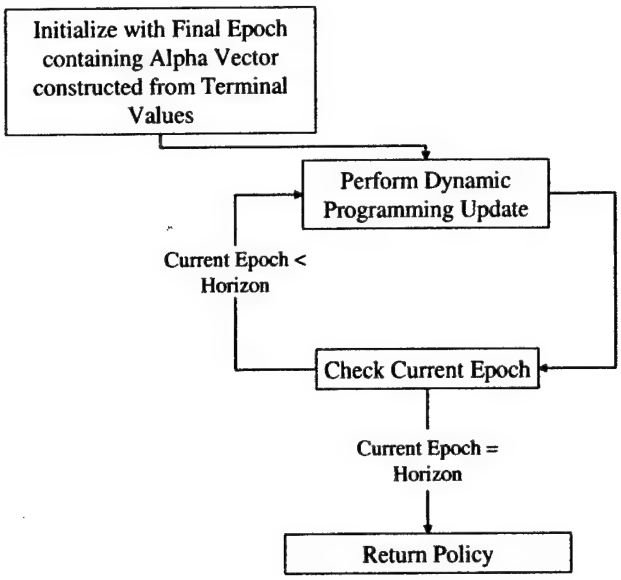

Figure 3-15: General finite-horizon POMDP algorithm framework

A clear method of dividing up POMDP solution algorithms is to first consider those that solve for the optimal value function and then consider those that solve for an approximate value function.

3.3.2.1 Exact Algorithms

Exact POMDP solution algorithms can be further classified by how they construct the value function. In general, all of the algorithms use $V_{k-1}(\pi)$ to build $V_k(\pi)$. However, this can be accomplished one of two ways, either through *enumeration* or *construction*.

Enumerative algorithms use the fact that in a finite horizon problem, there will only be a finite number of alpha vectors, $\alpha^{\psi}(k)$, that can be constructed based upon a finite list of actions, Ψ_i, and $V_{k-1}(\pi)$. Thus, it is possible to enumerate all alpha vectors for epoch k. Alpha vectors are constructed using (3.9) [10] for all extreme points of the belief space.

$$\alpha^{\psi}(k,s) = r_{sk}^{\psi} + \sum_{\theta \in \Theta_i,\, s' \in S} \mathcal{E}_{ss'}^{\psi} \mathcal{O}_{s'}^{\psi\theta} \alpha_u(k-1,s') \quad \forall \psi \in \Psi, \theta \in \Theta, u \in U(k-1). \qquad (3.9)$$

Equation (3.9) can generate a large number of alpha vectors. For a relatively small problem with 2 actions and 2 observations the number of generated alpha vectors explodes as the number of epochs increases. As stated earlier, $V_0(\pi)$ has only one alpha vector. Thus, the number of alpha vectors generated for epoch 1 will be $2*1^2$. Similarly, the number of alpha

vectors generated for possible inclusion in $V_2(\pi)$ will be $2*2^2$ which equals 8. Further results are listed in *Table 3-4*.

Epoch	1	2	3	4	5	6	7
Potential α	2	8	128	32768	2.15 E9	9.22 E18	1.70 E38

Table 3-4: Potential alpha vectors for problem with 2 actions and 2 observations

Generation and storage of this many alpha vectors is impossible for long horizons. A POMDP formulation of the targeting cycle problem that has tens or even hundreds of actions and many observations would drive these numbers even higher. Thus an algorithm is needed to reduce the set of generated alpha vectors to those in the *parsimonious set*, the set of alpha vectors that make up the value function as seen in *Figure 3-14*.

Monahan [31] was the first to propose such an algorithm, which he credited to Sondik. His reduction phase uses linear programming. The purpose of the linear program is to determine whether there is a point in the belief space such that the alpha vector under consideration, $\alpha^*(k)$, dominates all the other alpha vectors. In order to do this reduction phase, the following LP is set up for each alpha vector:

$$\max_{\pi} \quad 0 \tag{3.10}$$

$$\sum_{s\in S}\pi_s\left(\alpha(k,s)-\alpha^*(k,s)\right)\le 0 \qquad \forall \alpha \ne \alpha^* \tag{3.11}$$

$$\sum_{s\in S}\pi_s = 1 \tag{3.12}$$

$$\pi_s \ge 0 \qquad \forall s\in S. \tag{3.13}$$

It contains a set of domination constraints (3.11), a convexity constraint (3.12), and nonnegativity constraints (3.13). The need for a convexity constraint is clear because we wish to find a point in the belief space so the decision variables (i.e., probabilities) in question must sum to one. Nonnegativity constraints ensure all points in the belief space are positive. The decision variables, π_s, form a belief state.

While these constraints maintain feasibility in the belief space, the other constraints, (3.11), one constraint for all other alpha vectors except the one under consideration, are set up to find a point at which the alpha vector under consideration dominates. This is done by summing over all the states, the decision variable for that state times the difference between the values of the alpha vector for the constraint and the alpha vector under consideration at that state. If this

value is less than 0 we know that $\alpha^*(k)$ dominates the other alpha vector at a point in the belief space. The objective function is irrelevant as we seek a feasible solution. Such a point would indicate that the alpha vector under consideration does in fact belong in the parsimonious set. If this LP is infeasible for $\alpha^*(k)$, $\alpha^*(k)$ does not belong in the parsimonious set and thus can be discarded. As an added bonus, it does not have to be represented by a constraint in future checks.

While this algorithm will return the parsimonious set, there is the issue that arises when an alpha vector dominates at only one point. This occurs when three or more alpha vectors that are part of the parsimonious set meet at a single point as shown for the three alpha vectors in *Figure 3-16.*

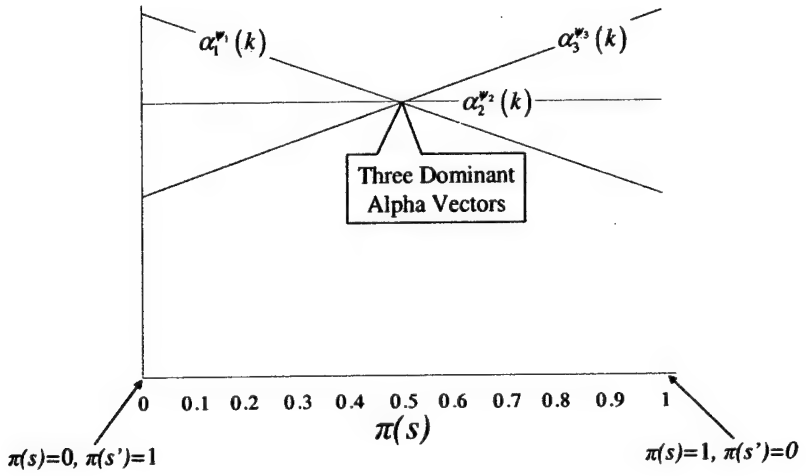

Figure 3-16: Multiple Alpha Vector Dominance at a Single Point

Some of these vectors do not provide a better solution than the others at such a point and thus are extraneous. A slight modification can be made to Monahan's algorithm to deal with such a case. This modification is explained in *Appendix B, Section B.2.1.1.*

Eagle [18] proposed an improvement upon Monahan's algorithm. Eagle recognized that setting up and "solving" a large number of LPs can be time-consuming. In order to reduce the number of LPs that must be solved, Eagle's modification checks each new alpha vector that is generated against the set of alpha vectors that have already been generated. If the new vector is dominated at all end points, thus component-wise dominated, by another alpha vector, it is discarded. Also, if a previously generated vector is component-wise dominated by the new

vector, the stored vector is replaced by the new vector. Beyond these steps, Eagle's algorithm works the same as Monahan's.

While the enumerative algorithms are simple to understand and implement, the enumeration phase, even with Eagle's additional checks, can produce an immense number of alpha vectors. Therefore, use of these algorithms should be limited to problems with a small number of actions, observations, and time steps.

A different approach is taken by *constructive algorithms*. Rather than building all possible alpha vectors and then paring down the set, constructive algorithms build the value function one alpha vector at a time. No extraneous alpha vectors are generated, thus substantially reducing the number that are built and have to be stored. This is done by using an equation similar to (3.9), except that it considers which alpha vector from the previous epoch to use in the summation rather than generating alpha vectors based on all of them. This consideration includes the belief state and action for which the alpha vector is being generated, and the current observation in the summation. Development and further explanation of these DP-like equations can be found in Cassandra [10].

$$\alpha^{\psi}(k,s) = r_{sk}^{\psi} + \sum_{\theta \in \Theta_j,\, s' \in S} \mathcal{E}_{ss'}^{\psi} \mathcal{O}_{s'}^{\psi\theta} \alpha_{\kappa(\pi,\psi,\theta)}(k-1,s'). \tag{3.14}$$

$$\kappa(\pi,\psi,\theta) = \arg\max_{u \in U(k-1)} \sum_{i,\, j \in S} \pi(i) \mathcal{E}_{ij}^{\psi} \mathcal{O}_{j}^{\psi\theta} \alpha_{u}(k-1,j). \tag{3.15}$$

Using (3.14) and (3.15), we can generate an alpha vector for action ψ for any point π in the belief space by calculating its value at the extreme points of the belief space. In order to find the optimal alpha vector for belief state π, $\alpha_{\pi}^{\psi^*}(k)$, we simply use equation (3.14) for all actions and pick the one that has the maximum value at π. Equation (3.16) shows the mathematical interpretation.

$$\alpha_{\pi}^{\psi^*}(k) = \max_{\psi \in \Psi_j} \left\{ \alpha^{\psi}(k) \cdot \pi \right\}. \tag{3.16}$$

Determining the points for which to generate alpha vectors is the key to the constructive algorithms of which Sondik proposed the first, the One-Pass algorithm. At its core, Sondik's One-Pass algorithm finds an alpha vector for a point in the belief space, determines the region over which the generated alpha vector dominates, and then checks points at the edge of this region for other dominant alpha vectors. Of these procedures, determining the region of dominance is the most interesting. To partition the belief space, Sondik uses linear

programming. Development of this LP formulation is quite in depth and the reader is referred to Sondik [34] and Cassandra [10] for full explanation. As noted by a number of authors, to include Mukherjee and Seth [32] and Cassandra [10], the One-Pass algorithm has some fundamental subtleties and flaws. In fact, as proposed by Sondik, the One-Pass algorithm is not guaranteed to find the optimal value function due to the LP formulation [32]. In addition, the algorithm chooses dominance regions conservatively and thus explores a large number of regions when in reality only a few need to be considered [10].

Building upon Sondik's foundation, Cheng [13] developed two algorithms for solving POMDPs. The first of these, the Relaxed Region algorithm closely follows the steps of the One-Pass algorithm. However, as the name indicates, the regions that this algorithm defines are larger than the regions defined in Sondik's approach, reducing the number of regions that must be considered. Making this change, however, requires a fundamental change in how the problem is solved. Rather than simply solving an LP, Relaxed Region requires that we find the extreme points of the defined region. There is no objective function value so a vertex enumeration method such as those described by Mattheiss [28] or Mattheiss and Rubin [29] can be used. Again, the development of the constraint set and the rationale behind these constraints are quite complex. Further discussion can be found in Cheng [13] or Cassandra [10].

Cheng showed that the Relaxed Region algorithm solved a POMDP more efficiently than the One-Pass algorithm due to the smaller number of regions defined and thus a smaller number of extreme points need to be enumerated and checked. Also, Cheng's formulation requires a smaller number of constraints. He was not, however, able to make the definitive theoretical statement that his algorithm solved POMDPs faster or with less memory than Monahan's enumeration algorithm. Recognizing the limiting factor in the algorithm was the complex constraint set necessary to define the belief space partitions, Cheng developed an algorithm that uses a much simpler constraint set.

From this idea came the Linear Support algorithm. Beyond the reduction in the complexity of the constraint set, the Linear Support algorithm has the attractive property of always having a lower bound approximation of the value function over the entire belief space. As opposed to other constructive algorithms, this allows the Linear Support algorithm to be stopped before the complete optimal value function is built and still have a valid policy over the entire belief space. Such a property is extremely useful as we model the targeting cycle.

The Linear Support algorithm begins by building the optimal alpha vectors for the extreme points of the belief space using equations (3.14), (3.15), and (3.16). The intersection of these alpha vectors is then determined. If the current value function at the point of intersection is not optimal, the optimal alpha vector is generated for that point, again using equations (3.14), (3.15), and (3.16). Intersections of $|S|$ alpha vectors are found, checked, and optimal alpha vectors generated if we do not have the optimal value function at that point. This cycle continues until the optimal value function has been found. A step by step explanation of this algorithm with accompanying graphs for a two dimensional problem can be found in *Appendix B, Section B.1.*

Cheng ran a number of empirical tests comparing his two algorithms against those of Monahan and Sondik. *Table 3-5* shows results for a slightly larger version of Sondik's machine inspection and replacement problem, which had 3 states, 4 actions, 2 observations, and a horizon of 20, showed the benefit of the Relaxed Region and Linear Support algorithm with respect to CPU time.

Algorithm	One-Pass	Enumeration with Eagle modification	Relaxed Region	Linear Support
CPU Time	2.947	0.937	0.894	0.751

Table 3-5: CPU time comparison of POMDP algorithms for a modified version of Sondik's machine inspection and replacement problem, Cheng (1988)

These results showed a marked improvement over previous algorithms. However, an even greater effect was found when the Linear Support algorithm was run as an approximate algorithm. This is done by slightly modifying the optimality check of the current value function approximation at a belief state. Rather than checking for equality between the current and optimal value functions, the difference is checked against an error tolerance, φ. If the difference is less than the error tolerance, this point is removed from the check list without generating a new alpha vector. In further numeric tests, on problems of varying size, Cheng showed how this modification can drastically reduce the solution time. Sondik's One-Pass algorithm was not considered due to Cheng's proof that both the Relaxed Region and Linear Support algorithms were superior and Monahan's Enumeration algorithm was run with Eagle's modification. Results from these empirical tests are shown in *Table 3-6.*

Algorithm	Enumeration	Relaxed Region	Linear Support φ=0	Linear Support φ=0.001	Linear Support φ=0.005	Linear Support φ=0.01	Linear Support φ=0.1
CPU Time	4.626	1.012	1.631	1.068	0.535	0.478	0.158
	5.171	2.230	2.348	1.267	0.727	0.692	0.389
	79.154	28.633	36.327	21.877	7.522	4.301	0.891
	762.173	N/A	172.282	57.226	9.140	2.604	2.152

Table 3-6: CPU time comparisons of POMDP algorithms for selected data, Cheng (1998)

These results show that allowing a small amount of error in the value function can greatly reduce the solution time. This fact will be used later in the decomposition algorithm.

To run the Linear Support algorithm, all extreme points of a convex polytope must be found. In lower dimensions this may seem somewhat trivial. However, in order to solve problems with varying state space sizes, this must be done in higher dimension. Such a problem has been researched in computational geometry and even determining the number of vertices has been found to be "...NP-hard in the strong sense" [17].

Recognizing this shortcoming, Littman, in conjunction with Cassandra and Kaelbling, developed the Witness algorithm [25]. While similar to the Linear Support algorithm, the Witness algorithm has two major differences: first, its approximation of the optimal value function and second, the way it identifies belief states that need to be checked. Rather than generating one approximation for the value function, the Witness algorithm generates multiple approximations, each based upon a single action. These approximations are then combined and the dominated alpha vectors are removed. Such an approach has an advantage in the case where an action dominates over a portion of the belief space and is represented by multiple alpha vectors. Details of how the individual approximations for each action are created and then combined can be found in Littman [25] or Cassandra [10]. These procedures involve linear programming formulations similar to those in the One-Pass and enumeration algorithms. While there is no theoretical evidence to show that the witness algorithm is better than Cheng's algorithms, "empirical results hint at this result" [10].

Currently, the state of the art in POMDP solution methods is actually a group of algorithms based upon the idea called *incremental pruning*. Algorithms of this type build up successive approximations of the value function based upon action-observation pairings in a similar manner to the Witness algorithm. However, the manner in which these approximations are constructed and subsequently reduced is quite different.

Originally proposed by Zhang and Liu [38] with a specific algorithm formalized by Cassandra, Littman, and Zhang [11], incremental pruning methods build a value function, $V_k^{\psi\theta}(\pi)$, for an action ψ and observation θ. Value functions of this type are combined to form an action-based value function, $V_k^{\psi}(\pi)$. Finally, the action-based value functions are combined to form the complete value function $V_k(\pi)$. Details of how $V_k^{\psi\theta}(\pi)$ is constructed as well as the reduction of dominated alpha vectors at each successive combination of approximate value functions can be found in *Appendix B, Section B.2.*

Again, theoretical results provide little insight into how well Incremental Pruning methods, and specifically Cassandra, Littman, and Zhang's variant called the Restricted Region (RR) algorithm, fare against other POMDP solution algorithms. To test their algorithm, Cassandra, Littman, and Zhang ran RR on a number of standard POMDP problems of varying size. The results in *Table 3-7* show that this new method empirically performs better than the witness algorithm.

	Total Run Time (sec.)		
POMDP Problem	Witness	Incremental Pruning with RR	Enumeration
4x3	727.1	157	N/A
4x4	3226	909.2	216.7
Cheese	351.8	203.3	N/A
Part Painting	5608.4	5226.4	1116.9
Network	6422.9	722.5	N/A

Table 3-7: Computation times for selected POMDP algorithms on classic problems, Cassandra, Littman, and Zhang (1997). Full explanations of the individual problems can be found on Cassandra's POMDP website, <http://www.cs.brown.edu/research/ai/pomdp/examples/index.html>. Note that when enumeration works, it does better than either of the other two algorithms.

3.3.2.2 Approximate Algorithms

Even with advanced algorithms such as Incremental Pruning, solution times for POMDPs can be prohibitive, especially when contrasted to the quickly changing landscape of the modern battlefield. Problems such as the Network problem shown in *Table 3-7* are of reasonable size: 7 states, 4 actions, 2 observations, and a horizon of 14. In solving POMDPs for the targeting cycle problem, we might solve many problems of this size. However, if each takes even half the time required for the Network problem, a plan will be out of date before it is generated. A reasonable next step is to consider approximate algorithms. We have already covered such an algorithm in

Linear Support. Tuning the error tolerance allows for much faster solution times while maintaining a good approximation of the value function.

Another approximation approach is to divide the belief space with a grid and generate the alpha vectors for specific points in this grid. How the grid is constructed is what distinguishes algorithms of this type such as those proposed by Lovejoy [27], Hauskrecht [22], and Zhou and Hansen [39]. These algorithms are convenient because the amount of computational effort used to solve the problem can be specified. This stems from the fact that for each point in the grid, equation (3.16) is used to find the optimal alpha vector for that point. With the ability to specify computation restrictions, we lose the ability to specify an error tolerance for the value function. Interpolating between the value function at grid points gives an estimate of the error between the value function approximation and the true value function at any point in the belief space. As long as the extremes of the belief space are included in the grid, this interpolation will be an overestimate; this fact stems from the piecewise linear and convex properties of the value function. *Figure 3-17* illustrates interpolation for a two state problem. The end points of the interpolation are the value function values at the selected grid points.

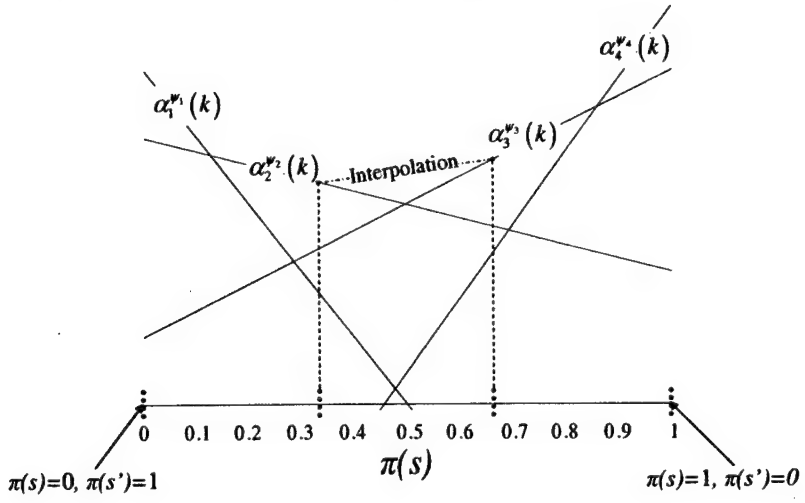

Figure 3-17: Interpolation for Grid Based POMDP Solution Algorithms

Lovejoy (1991) shows that his interpolation and grid selection scheme maintain this upper bound property when dynamic programming is applied.

3.3.3 Targeting Cycle POMDP Models and Hierarchy

With the background knowledge of POMDPs and their solution methods, we now formulate the POMDP models for the targeting cycle problem.

- **Set Definitions and Common Data**

  Object index: $i \in I$
  States: $s \in S$
  Allowable action set for object i: $\psi \in \Psi_i$
  Contact i possible types: $\xi \in \Xi_i$ (i.e., tank, SAM, SSM, etc.)
  Possible observations for object i: $\theta \in \Theta_i$
  Horizon: T
  Epoch: $k \in \mathbb{Z}^+ \leq T$
  Number of aircraft a used by action ψ: AFTUSE_a^{ψ}
  Number of weapons w used by action ψ: WPNUSE_w^{ψ}
  Number of sensors b used by action ψ: $\mathrm{SENSUSE}_b^{\psi}$

  All action lists contain a "Pause" action that uses no resources, does not influence the object being acted upon, and has a uniform observation model.

- **Area of Interest POMDP Input Data**

  S={0, 1, 2, ...}
  Θ ={0, 1, 2, ...}
  Probability action ψ will discovery an object when applied to area i: PDIS_i^{ψ}
  Probability of attrition of aircraft type a under action ψ applied to area i: PA_{ai}^{ψ}
  Average value of objects in area i: AVGVAL_i
  Cost of action ψ at epoch k applied to area i: C_{ik}^{ψ}

  It is important to note that the discovery probability, PD_i^{ψ}, is for each object in area i, independent of the other objects in area i.

- **Contact POMDP Input Data**

  $S = \Xi_i$
  $\Theta = \Xi_i$.
  Probability contact i will evade when action ψ is applied: EV_i^{ψ}
  Probability of receiving observation θ when action ψ is applied to contact i when it is in state s: $\mathcal{O}_s^{\psi\theta}$
  Probability of attrition of aircraft type a under action ψ applied to contact i: PA_{ai}^{ψ}
  Value of possible type ξ for contact i: AVGVAL_i^{ξ}
  Cost of action ψ at epoch k applied to contact i: C_{ik}^{ψ}

If a contact evades, it is assumed to move to the "Not a target" state and thus is essentially lost. It is assumed that the contact will not evade if a "Pause" action is taken. The action list for contacts includes "Declare" actions that indicate a level of certainty that the contact is of a certain type. It is from these "Declare" actions that value is attained. Further explanation of the "Declare" actions can be found in *Chapter 4, Section 4.2.*

- **Target POMDP Input Data**

  S={Live, Dead}
  Θ={Live, Dead}
  Probability of kill for action ψ applied to target *i*: $\mathcal{E}_{LD}^{\psi}$
  Probability of receiving observation θ when action ψ is applied to target *i* when it is in state *s*: $\mathcal{O}_{s}^{\psi\theta}$
  Probability of attrition of aircraft type *a* under action ψ applied to target *i*: PA_{ai}^{ψ}
  Probability of target *i* moving from dead state to live state due to repair when action ψ is applied: $\mathcal{E}_{DL}^{\psi}$
  Value of target *i*: VAL_{i}
  Cost of action ψ at epoch *k* applied to target *i*: C_{ik}^{v}

  The repair probability, $\mathcal{E}_{DL}^{\psi}$, is assumed to be zero for relocatable targets.

In modeling the targeting cycle, these POMDP models interact in a hierarchy. Beyond the objects that are designated at the outset of the problem by intelligence preparation of the battlefield (IPB), objects might be discovered in an area of interest, identified, and then struck if the object is determined to be a valid target. Over time, objects move down this hierarchy until they are placed in the "Not a Target" box for relocatable targets and objects determined not to be valid targets and into the "Fixed Target" box for fixed targets. *Figure 3-18* illustrates the POMDP hierarchy in which IPB initializes objects in each POMDP category. Over time, objects are discovered and are dealt with through a contact POMDP. Once they have been identified as a relocatable target, a fixed target, or not a target, with a high enough certainty, above x%, they are moved accordingly. While fixed targets are never completely destroyed due to the repair assumption from *Chapter 2, Section 2.3*, relocatable targets are assumed not to regenerate and thus can be moved into the "Not a Target" box once the belief that the target is dead is sufficiently high, above y%.

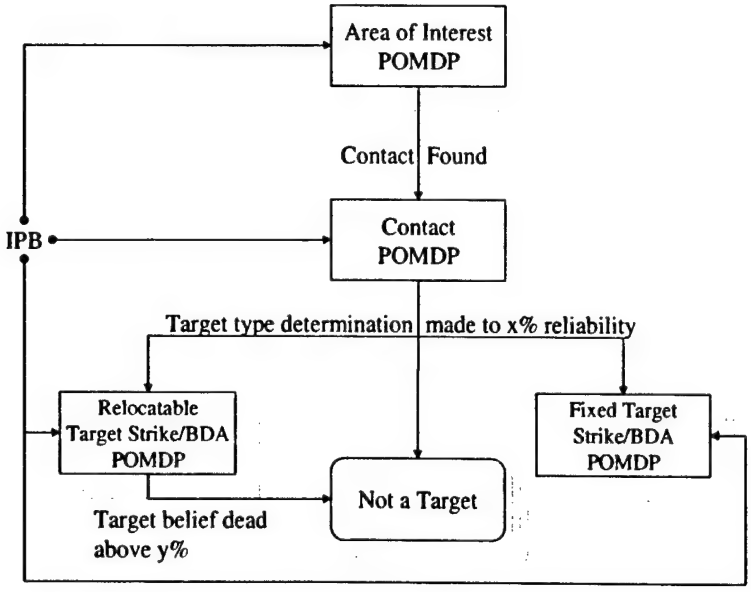

Figure 3-18: POMDP hierarchy. Over time objects progress down the hierarchy as they are discovered, identified, and struck, if appropriate.

3.4 Chapter Summary

This chapter began with an overview of the characteristics and assumptions of the targeting cycle problem and how different modeling techniques have been applied to solve problems similar to the targeting cycle problem. Resource allocation methods address most aspects of the targeting cycle problem but fall short due to the large number of possible contingency plans. Policy development methods address this problem but deal with action costs rather than resource constraints. A hybrid approach, as proposed by Yost, combines these two in order to adequately solve the sensor-shooter problem and thus we extend that technique to address the targeting cycle problem. In order to lay the groundwork for this extension, we covered the major POMDP solution algorithms and then defined the POMDP models for the three types of objects we will be dealing with: areas of interest, contacts, and targets.

4 Resource and Task Assignment

Having explored the POMDP model, and the versatility it provides in modeling a stochastic environment, we can further discuss the interaction between the resource assignment LP and the task assignment POMDPs. We have seen that a hybrid approach such as Yost's [37] is a good model for the sensor-shooter problem. As shown in *Chapter 3*, *Section 3.3.3*, further POMDP models can be developed to expand this formulation to include *discovery in an area of interest* and *contact identification*, thus extending the sensor-shooter problem to the targeting cycle problem.

4.1 Motivation

The targeting cycle, as described in *Chapter 2*, is complex and fast-paced. Currently, Air Force Air Operations Centers (AOC) need one person per planned sortie to make sure everything gets done [19]. This is an extremely high number when you consider that a typical air tasking order (ATO) can contain a thousand or more sorties. As Lt. Gen. Ronald E. Keys said "We need technology to do all those repetitive and cataloging tasks so we don't need as many people..." [19]. Repetitive tasks such as assuring resource availability and deconflicting resource usage between different sorties are two such tasks that the LP/POMDP hybrid approach handles.

In addition, the LP/POMDP formulation of the targeting cycle problem provides planners with dynamic plans that account for the stochastic nature of the world. Although future resource

usage is considered in expectation, the inclusion of possible future observations, and the effects these observations have upon military planning, helps make the plans robust against real world events. Another key element of this formulation is the streamlining of the interactions between intelligence and operations. With explicit models of ISR actions included, the LP/POMDP formulation of the targeting cycle problem can be updated and re-run as events transpire and observations are received.

4.2 LP/POMDP Formulation and Algorithm

Formulating the targeting cycle problem using the LP/POMDP hybrid approach necessitates a few additions to Yost's formulation of the sensor-shooter problem [37]. One reason that Yost was able to address problems of such large size was his grouping of targets into target types, the assumption being that all targets of the same type are represented in the model with the same value and the same allowable actions. Our reduction of the target type set to an individual target provides more flexibility with respect to target valuation and allowable actions but reduces the total number of targets that can be considered. We also formulate POMDPs for individual contacts and areas of interest. Thus, in solving the POMDP sub-problems, there will be a POMDP for each target, contact, and area of interest.

In order to start the algorithm, we need to give the master LP some contingency plans with which to generate dual values. A contingency plan of all pauses is generated for all objects to ensure that the LP is always feasible. For targets, the three best single-shot contingency plans are generated for all time steps and we generate the three best shoot-look-shoot contingency plans beginning in all time steps except T-1 and T. We only use ISR resources when dealing with contacts and areas of interest; thus, we generate the best three single-look contingency plans for all time steps and the best three double-look contingency plans beginning in all time steps except T. When these contingency plans have been generated, their associated resource usage and expected rewards are calculated and loaded into the LP, which is then solved, yielding dual prices for the available resources. These dual prices are passed to the POMDP sub-problems to use in the calculation of C_{ik}^{ψ}, the cost of applying action ψ against object i during epoch k. These POMDPs are then solved, providing a policy from which an improving contingency plan is constructed. The new contingency plan is used to generate a column which is then added to the LP.

We make a simplifying change in the stopping criteria for this algorithm in that we stop when the change in the objective function and the maximum change in dual values is less than the numeric error tolerance, ε. The primal objective function value is commonly used as part of the stopping criteria in column generation algorithms. The duals are a direct input into the POMDP sub-problems through the action costs, thus we need to consider changes in the individual rather than their conglomeration in the dual objective function value when deciding to terminate the algorithm. *Figure 4-1* is a graphical representation of the LP/POMDP algorithm for the targeting cycle problem.

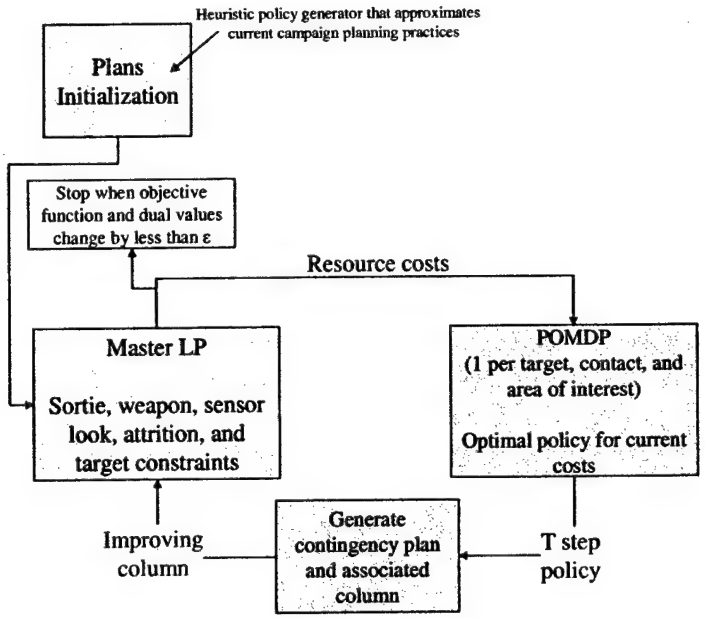

Figure 4-1: LP/POMDP algorithm for targeting cycle problem

With this structure in mind we can construct the master LP that performs the resource and task assignment.

- **Set Definitions and Common Data**

  Aircraft types: A
  Weapons types: W
  ISR sensor types: B
  Object set: I
  Area of interest set: $\mathcal{A} \subseteq \mathrm{I}$
  Contact set: $\mathcal{U} \subseteq \mathrm{I}$
  Target set: $\mathcal{T} \subseteq \mathrm{I}$
  Contingency plans for object i: O_i
  Contact i possible types: Ξ_i

Horizon: T
Time step: $t \in \mathbb{Z}^+ \leq T$

- **Input Data**

  Average value of objects in area of interest *i*: $EVAL_i$
  Value of identifying contact *i* as type ξ given that it is of type ξ': $IDVAL_{\xi\xi'i}$
  Value of target *i*: VAL_i
  Number of aircraft of type *a* available at time *t*: NAA_{at}
  Number of ISR sensors of type *b* available at time *t*: ISR_{bt}
  Number of weapons of type *w* available: WPN_w
  Maximum allowable attrition of aircraft type *a*: $MAXATTA_a$
  Belief that contact *i* is of type ξ: $PT_{\xi i}$
  Belief that target *i* is dead: PD_i

- **Contingency Plan Data**

  Expected number of aircraft of type *a* needed to prosecute contingency plan *o* against object *i* in time period *t*: NA_{aoit}
  Expected number of ISR sensors of type *b* needed to prosecute contingency plan *o* against object *i* in time period *t*: LKS_{boit}
  Expected number of weapons of type *w* needed to prosecute contingency plan *o* against object *i*: WE_{woi}
  Expected attrition for aircraft type *a* under contingency plan *o* against object *i*: $ATTA_{aoi}$
  Expected number of discoveries in area of interest *i* under contingency plan *o*: $EDIS_{oi}$
  Probability of declaring contact *i* as type ξ after applying contingency plan *o*: $PDEC_{\xi oi}$
  Expected belief that target *i* is dead after applying contingency plan *o*: ED_{oi}

- **Decision Variables**

  Proportion of contingency plan *o* to apply to object *i*: x_{oi}

- **Objective Function**

$$\max_x \sum_{i \in A} \sum_{o \in O_i} EVAL_i \, EDIS_{oi} \, x_{oi} + \sum_{i \in U} \sum_{o \in O_i} \sum_{\xi \in \Xi_i} \sum_{\xi' \in \Xi_i} PDEC_{\xi oi} \, IDVAL_{\xi\xi'i} \, PT_{\xi' i} \, x_{oi} + \sum_{i \in T} \sum_{o \in O_i} VAL_i \left(ED_{oi} - PD_i \right) x_{oi} \tag{4.1}$$

- **Constraints {Dual Information}**

$$\sum_{i \in I} \sum_{o \in O_i} NA_{aoit} x_{oi} \leq NAA_{at} \quad \forall a \in A, t \leq T \qquad \{ad_{at}\} \tag{4.2}$$

$$\sum_{i \in I} \sum_{o \in O_i} LKS_{boit} x_{oi} \leq ISR_{bt} \quad \forall b \in B, t \leq T \qquad \{ld_{bt}\} \tag{4.3}$$

$$\sum_{i\in I}\sum_{o\in O_i} \mathrm{WE}_{\mathrm{woi}} x_{oi} \le \mathrm{WPN}_{\mathrm{w}} \quad \forall w \in W \qquad \{\mathrm{wd}_{\mathrm{w}}\} \qquad (4.4)$$

$$\sum_{i\in I}\sum_{o\in O_i} \mathrm{ATTA}_{\mathrm{ao}} x_{oi} \le \mathrm{MAXATTA}_{\mathrm{a}} \quad \forall a \in A \qquad \{\mathrm{am}_{\mathrm{a}}\} \qquad (4.5)$$

$$\sum_{o\in O_i} x_{oi} = 1 \quad \forall i \qquad \{\mathrm{td}_{\mathrm{i}}\} \qquad (4.6)$$

$$0 \le x_{oi} \le 1 \quad \forall i \in I, o \in O_i . \qquad (4.7)$$

The objective function value of this LP maximizes the expected reward of selected contingency plans. For areas of interest, benefit is gained by discovering objects, while correctly identifying a contact yields benefit. Benefit is gained with respect to targets when the target is transitioned from the live state to the dead state. Equation (4.2) ensures that the expected number of aircraft used does not exceed the number available for all the different aircraft types in all periods. Equation (4.3) does the same but for ISR sensors. Feasible expected weapons usage is maintained by (4.4) while (4.5) ensures that we do not risk too much attrition. The requirement to act upon every object is enforced by (4.6), as (4.7) keeps the proportional plan usage within the admissible bounds.

With this information, we can now determine the action costs and associated rewards for the POMDPs. C_{ik}^{ψ} is found using the following equation and the action data as defined in *Chapter 3, Section 3.3.3*:

$$C_{ik}^{\psi} = \sum_{a\in A} \mathrm{AFTUSE}_{\mathrm{a}}^{\psi} \mathrm{ad}_{\mathrm{a\ T\text{-}k+1}} + \sum_{w\in W} \mathrm{WPNUSE}_{\mathrm{w}}^{\psi} \mathrm{wd}_{\mathrm{w}} + \sum_{b\in B} \mathrm{ISRUSE}_{\mathrm{b}}^{\psi} \mathrm{ld}_{\mathrm{b\ T\text{-}k+1}} + \sum_{a\in A} \mathrm{AFTUSE}_{\mathrm{a}}^{\psi} \mathrm{PA}_{\mathrm{ai}}^{\psi} \mathrm{am}_{\mathrm{a}} \quad . \qquad (4.8)$$

A POMDP will incur this immediate cost when action ψ is used at epoch k, which corresponds to time step T-k+1.

Value is input for area of interest and target POMDPs through the 0-epoch alpha vector. For an area of interest i, with a maximum possible number of objects S_{max}, the end state values for the 0-epoch alpha vector are:

$$\alpha(0,s) = (\mathrm{S}_{\max} - s)\mathrm{EVAL}_{\mathrm{i}} \quad \forall s \in S . \qquad (4.9)$$

Targets in the live state are assumed to have a value of zero, thus the 0-epoch alpha vector for target i is:

$$\alpha(0, live) = 0; \text{ and} \tag{4.10}$$

$$\alpha(0, dead) = \mathrm{VAL}_i. \tag{4.11}$$

Contact POMDPs, however, do not receive terminal value through 0-epoch alpha vectors. Rather, value is gained by declaring a contact as a certain type. Thus, the allowable action list for contact i will include "Declare" actions that use no resources and have end state values:

$$\alpha^{Declare\,\xi}(k, \xi') = \mathrm{IDVAL}_{\xi\xi'_i} \quad \forall \xi, \xi' \in \Xi_i. \tag{4.12}$$

With this general layout of the targeting cycle LP/POMDP formulation, we can address some of the problem-specific assumptions.

4.3 Initialization Techniques

There might not be the intuition that dictates what kinds of contingency plans should be used to initialize the LP when running this decomposition as a proof-of-concept,. In fact, the whole purpose of a proof-of-concept is to develop the basic understanding of how the system will work.

Rather than starting the algorithm at the LP as illustrated in *Figure 4-1*, the algorithm can be started at the POMDP sub-problems, which might be more appropriate and computationally effective. However, this requires dual prices for aircraft, weapons, sensors, and aircraft attrition. Generating such values *a priori* is similar to the problem of generating "good" contingency plans, which might be difficult and provide little insight for the problem. A simpler approach may be taken: use an LP for which we know all the inputs and which has constraints similar to those in the master LP to generate the duals. This simplified LP takes a form similar to that of the master LP with the appropriate sets and data the same as the master LP and the POMDP sub-problems. We generate all possible action-object combinations and maximize a similar objective function for a one-step problem. *Figure 4-2* illustrates the LP/POMDP algorithm for the targeting cycle problem with dual initialization.

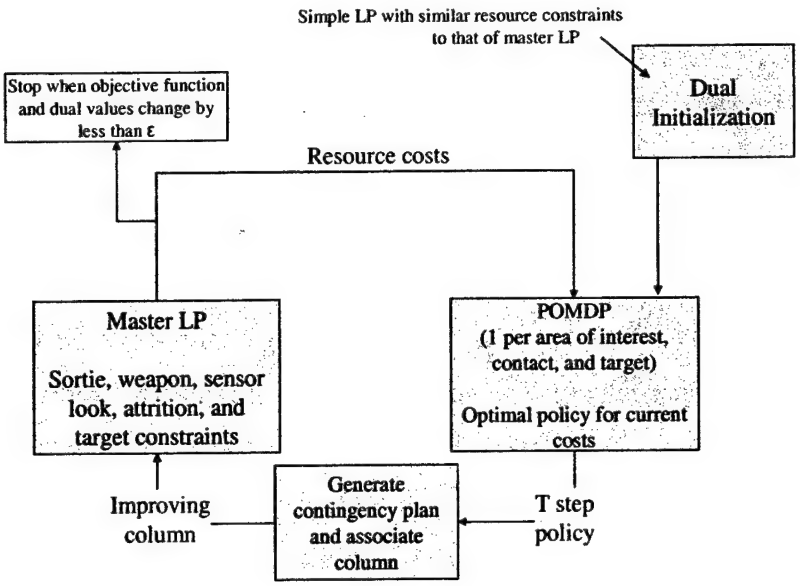

Figure 4-2: LP/POMDP algorithm for targeting cycle problem with dual initialization

- **Input Data**

  Current belief that area *i* is in state *s*: PN_{si}
  Weighting factor between strike and BDA actions for targets: λ

- **Decision Variables**

  Proportion of action ψ to apply to object *i*: $x_{\psi i}$

- **Objective Function**

$$\max_{x} \sum_{i\in\mathcal{A}}\sum_{\psi\in\Psi_i}\sum_{s\in S} \mathrm{EVAL}_i \mathrm{PDIS}_i^{\psi} \mathrm{PN}_{si} |s| x_{\psi i} +$$
$$\sum_{i\in\mathcal{A}}\sum_{\psi\in\Psi_i}\sum_{\xi\in\Xi_i}\sum_{\xi'\in\Xi_i} \mathrm{PT}_{\xi i} \mathcal{O}_{\xi'}^{\psi\xi} \mathrm{IDVAL}_{\xi\xi'} x_{\psi i} + \qquad (4.13)$$
$$\sum_{i\in\mathcal{T}}\sum_{\psi\in\Psi_i} \mathrm{VAL}_i \left[\left(\lambda[1-\mathrm{PD}_i]\mathcal{E}_{LD}^{\psi}\right) + (1-\lambda)\left([1-PD_i]\mathcal{O}_L^{\psi L} + \mathrm{PD}_i\mathcal{O}_D^{\psi D}\right)\right] x_{\psi i}$$

- **Constraints {Dual Information}**

$$\sum_{i\in I}\sum_{\psi\in\Psi_i} \mathrm{AFTUSE}_a^{\psi} x_{\psi i} \le \mathrm{NAA}_{a1} \quad \forall a\in A \qquad \{\mathrm{ad}_{at}\} \qquad (4.14)$$

$$\sum_{i\in I}\sum_{\psi\in\Psi_i} \mathrm{SENSUSE}_b^{\psi} x_{\psi i} \le \mathrm{ISR}_{b1} \quad \forall b\in B \qquad \{\mathrm{ld}_{bt}\} \qquad (4.15)$$

$$\sum_{i\in I}\sum_{\psi\in\Psi_i} \mathrm{WPNUSE}_w^{\psi} x_{\psi i} \le \mathrm{WPN}_w \quad \forall w\in W \qquad \{\mathrm{wd}_w\} \qquad (4.16)$$

$$\sum_{i\in I}\sum_{\psi\in\Psi_i} \mathrm{AFTUSE}_a^{\psi} \mathrm{PA}_{ai}^{\psi} x_{\psi i} \le \mathrm{MAXATTA}_a \quad \forall a\in A \qquad \{\mathrm{am}_a\} \qquad (4.17)$$

$$\sum_{\psi \in \Psi_i} x_{\psi i} = 1 \quad \forall i \qquad \{td_i\} \qquad (4.18)$$

$$0 \le x_{\psi i} \le 1 \quad \forall i \in I, \psi \in \Psi_i. \qquad (4.19)$$

The objective function, (4.13) can be divided into three parts: one for areas of interest, another for contacts, and the third for targets. The first part of the objective function calculates the reward gained from the expected number of discoveries for each action applied to each area of interest. The second set of terms calculates the expected reward for the identification of each contact, while the final term calculates a weighted sum of strike and BDA reward for each action applied to the set of targets. Equation (4.14) ensures that the expected number of aircraft used does not exceed the number available for all the different aircraft types for the first period. Equation (4.15) does the same, but for ISR sensors. Feasible expected weapons usage is maintained by (4.16), while (4.17) ensures that we do not risk too much attrition. The requirement to act upon every object is enforced by (4.18) as (4.19) keeps the proportional action usage within the admissible bounds.

Once this LP is solved, the action costs for the POMDPs can be calculated using equation (4.8) and the POMDPs solved to generate an initial set of contingency plans. To ensure feasibility, as in the original formulation, the master LP begins with one contingency plan for each object that pauses for all time steps and thus uses no resources. The new columns associated with the generated contingency plans are added to the master LP, which is then solved, generating new duals. As in the original formulation, these iterations will continue until the change in the objective function and the maximum change in the duals are less than ε. Computational comparisons between the two types of initialization procedures are explored in *Chapter 5*. In general, we cannot say which will be faster but rather presume that the initialization procedure is an individual choice depending upon familiarity with the problem at hand.

4.4 Implementing Targeting Cycle Characteristics and Assumptions

While maintaining the general structure of Yost's algorithm, there are different assumptions for the targeting cycle that must be considered.

4.4.1 Mobile Contacts

As explored in Chapter 2, a contact might not be stationary. Rather, much like the Scud missile launchers during Desert Storm, a contact might be stationary for a period, perform some mission at that location, and then quickly move to a "safe" location. A second case is that a contact might move when its position is compromised. While this is unlikely to happen when a Global Hawk or another high altitude ISR platform finds it in an area of interest, it is a definite possibility when an easily observable aircraft, such as Predator, performs additional ISR missions. Modeling the first of these cases requires knowledge about the mission of a contact. Given that we have yet to determine the contact's object type, determining the specific mission of that contact is unlikely.

The second case, however, can be modeled. An estimated value for the percentage of the time that contacts move following an ISR action is performed can be found using means such as human intelligence or past experience. EV_i^{pause} is assumed to be zero but EV_i^{ψ}, as defined in *Chapter 3, Section 3.3.3*, for other actions can be determined. This transition probability is assumed to be independent of the contact's true type. While not considered in this work, a more detailed model could be developed such that the state-dependent value, $EV_i^{\psi}(s)$, is determined for a contact. This would help incorporate current information about the state of the contact in determining how likely a contact is to evade.

4.4.2 Fixed/Regenerative Targets

Fixed targets such as runways, bridges, and command and control (C2) facilities are important, reparable military assets. When damaged or destroyed, an adversary will work to get these assets functional as soon as possible. Since the advent of aerial bombing, the subject of regenerative targets has been studied. Today, we have extensive tables listing the times necessary to repair certain types of regenerative targets. These, of course, are estimates. An adversary may do nothing to rebuild an airfield or they may use all available resources. Under these scenarios, and everything in between, the time until military operations are restored can vary widely.

While targets such as tanks, artillery, and other such mobile military assets can be repaired when slight damage is inflicted, our choice to model all targets as either live or dead removes this possibility. If a tank is destroyed, it is more economically viable to build a new

tank than to repair the destroyed one. This, however, is not the case for fixed targets. Using the airfield example, building a new airfield could take months. Repairing a runway that has been hit by an anti-runway bomb, such as the BLU-107 Durandal or a conventional MK80 series bomb, can take a few days or even a few hours. Repairing such assets allows the enemy to quickly reengage.

To model the transition of regenerative targets from the dead state to the live state, $\mathcal{E}^{\psi}_{DL}\ \forall \psi \in \Psi_i$, we assume the transition follows a geometric distribution with the probability of success equal to the length of a time step divided by the mean repair time for the given target type. This approximation requires that the time steps be shorter than the expected repair time. Such an assumption is reasonable because of the relatively short time periods needed in the targeting cycle.

4.4.3 Problem Data

In solving the POMDP sub-problems, we rely upon many pieces of exogenous data that indicate the effectiveness of weapons and sensors. This data must be statistically estimated from historic data or derived from resource characteristics. To get an accurate estimate of the true system parameters, either of these methods, or a combination of the two, must be done in an appropriate manner based upon a significant amount of data. We make two assumptions about these data in order to use it in our decomposition: unbiasedness and small variance.

The trade-off between bias and variance has long been studied in statistics and econometrics. There are results from these fields that dictate how data is to be collected and analyzed in order to guarantee certain properties. In our case, biased estimates of system parameters would severely impact how the POMDPs determine the dominant actions over portions of the belief space. While this requirement is straightforward, having a low variance might not be. We can consider the performance estimate of a parameter as the mean of the observed distribution associated with that weapon or sensor. The variance indicates us how much the performance of the weapon or sensor deviates from the true value.

In comparing actions, the POMDPs simply look at the cost/reward of an action and the estimated parameters for that action. For example, if two strike actions that do not perform ISR have the same costs but one has a slightly higher probability of kill for the target type, that action will be chosen. This may be somewhat misleading. Suppose the munition had an idiosyncrasy

that caused its effectiveness to vary wildly. Such an example is that of laser guided munitions which perform superbly under clear weather conditions but do not perform well under adverse weather conditions. Knowing the mean value of its probability of kill does not give us the whole picture. Now suppose the other action uses a GPS guided bomb, which is somewhat less accurate in the best of conditions but it maintains this level of accuracy during adverse weather. In such a case the GPS munition might be preferable due to its lower variance, even though it has a lower probability of kill. In this work we assume that the variance of resource performance characteristics are small enough so as to have little affect upon deciding which action to choose. Formulations that model the variance of actions would be an important extension of this work.

4.4.4 Integer Solutions

When the LP/POMDP algorithm has converged to the optimal solution, as described in *Section 4.2*, we must solve the LP once more with integer constraints. This is necessary because we cannot use fractional resources. In all but special cases, to meet the resources constraints, equations (4.2) to (4.5), while still fully acting upon an object, equation (4.6), the optimal LP solution contains fractional decision variables. An integer programming (IP) or mixed integer programming (MIP) formulation is necessary. Yost addresses this problem by proposing a MIP in which decision variables corresponding to contingency plans with an initial action of pause are allowed to be fractional. In such a formulation, multiple contingency plans beginning with pause could be combined such that their total usage is integer. Contingency plans that begin with an action that consumes resources, however, are required to be integer.

While this formulation may initially seem to make sense, further examination brings up some interesting questions. Why is it that only contingency plans with initial pause actions are allowed to be combined? What does the combination of contingency plans signify and is it valid? While the first of these questions yields a general answer, the second is a philosophical question that does not seem to have a definite answer. Only allowing contingency plans with initial pause actions to be combined does not seem reasonable. Fundamentally, there is nothing different about contingency plans with initial pause actions and those with an initial action that uses resources. Thus, we propose a formulation in which contingency plans can be combined as long as they have the same initial action. Equations (4.20), (4.21), and (4.22) are used in place of equations (4.6) and (4.7).

Let x_{oi}^{ψ} be the decision variable representing a contingency plan o for object i that has an initial action of ψ and let y_i^{ψ} be the binary variable for contingency plans for object i that begin with action ψ. Also, the objective function, equation (4.1), is maximized over both x and y.

$$\sum_{o \in O_i} x_{oi}^{\psi} = y_i^{\psi} \quad \forall \psi \in \Psi_i, i \in I \tag{4.20}$$

$$\sum_{\psi \in \Psi_i} y_i^{\psi} = 1 \quad \forall i \in I \tag{4.21}$$

$$y_i^{\psi} \in \{0,1\}. \tag{4.22}.$$

This formulation will allow different contingency plans with the same initial action to be chosen at partial levels. Yost's approach would simply limit the y_i^{ψ} variables to cases in which ψ is the pause action and equations (4.6) and (4.7) would apply to all other decision variables.

Combinations of contingency plans might not make sense in a given problem and thus a third, fully integer, formulation is necessary. Such a formulation will force contingency plans to be fully chosen or not chosen at all. In order to do this, the following set of (4.23) replaces equation (4.7).

$$x_{oi} \in \{0,1\} \quad \forall o \in O_i, i \in I. \tag{4.23}$$

Solution characteristics for these three different IP/MIP formulations, such as solution time, objective function value, LP-IP/MIP gap, and selected contingency plans, will be explored in *Chapter 5*.

4.4.5 Accelerating POMDP Solutions

As observed by Yost [37], the POMDP sub-problems consume a large portion of the solution time. In some of our initial runs of a realistic problem, solution times ranged from tens of minutes to hours, with the vast majority of the time being spent solving the POMDP sub-problems. Yost suggests techniques for speeding up the sub-problems and thus the decomposition.

- **φ-Control**

The first technique suggested by Yost is to vary the POMDP error tolerance through *φ-control*. A classic problem with this type of "price-directive" decomposition was described by Dantzig in 1963 [15]. During the initial iterations of the algorithm there are severe swings in the

resource prices. Initially, the master LP duals are low and the sub-problems try to use a large amount of resources, pushing the price exceedingly high. Due to the high resource prices, the sub-problems produce columns that use very little resources, starting the cycle over again. This initial oscillation dampens as the master LP gets more columns to consider and balances between the high and low consumption columns.

In order to moderate this oscillation and spend less time solving initial POMDP sub-problems, we solve them with a loose error tolerance. As the algorithm converges, we systematically tighten the error tolerance until it reaches a specified lower bound. For the targeting cycle problem, we start with an initial φ and reduce it by a factor of ten each time the objective function and dual values change by less than ε, that is:

$$\varphi' = 0.1 \cdot \varphi . \tag{4-24}$$

This continues until φ is less than or equal to ε and the other stopping criteria are met. The computational impact of φ-control is examined in *Chapter 5*.

- **Action Control**

To calculate the optimal value at a point in the belief space using equation (3.8), we must maximize over all of the allowable actions for object *i*. Computationally, this is done by enumerating the function for all actions and picking the best one. This suggests that another way to speed up the POMDP sub-problems is to limit the number of actions they considered. However, we do not want to remove them from consideration altogether because this could lead to a sub-optimal solution. Therefore, we consider a method in which we limit the allowable actions for a number of iterations. At regular intervals, all of the actions are considered and the new allowable actions set is derived.

In solving the targeting cycle problem, we are solving a sequence of POMDPs. Thus, we can use information from previous POMDPs solutions to determine which actions to consider in future POMDPs. Specifically, we only consider those actions used in the policy of a previous POMDP. For each object, we solve the initial POMDP using all of the actions. We then limit Ψ_i to the actions used in the POMDP policy. This limitation on Ψ_i is maintained for a number of iterations based upon an *update interval*. At integer multiples of the update interval, all of the actions are considered and the new limited Ψ_i is derived from the policy generated. In *Chapter 5*, we consider the computational effects of different update intervals.

4.4.6 Rolling Horizon

Another important aspect of the targeting cycle problem is the rolling horizon, that is, we will only execute one time step of each selected contingency plan and then replan. There are two reasons such a technique is necessary. First, we do not know the actual length of the military campaign for which we are planning. Thus, we plan for a sufficiently long horizon so as not to use resources greedily in the beginning. Secondly, once a time step has been completed, we want to use the intelligence information that was gathered during that time step. This characteristic allows us to combine contingency plans as described in *Section 4.4.4*.

4.5 Solution Techniques

There are two main types of problems, the master LP, *Section 4.2*, and the subproblem POMDPs, *Chapter 3*, *Section 3.3.3*, that we need to solve, as well as to determine the interactions between these two problems. *Figure 4-3* shows the interactions between these two types of problems.

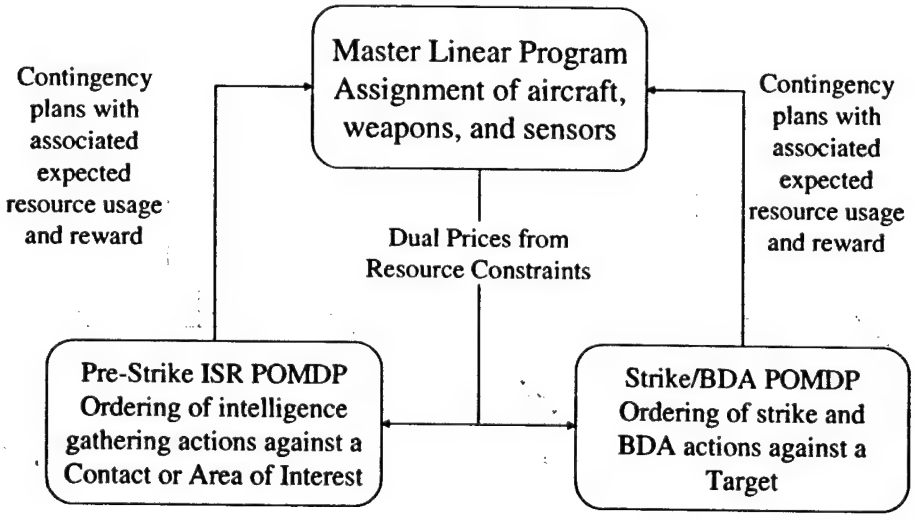

Figure 4-3: Hierarchical decomposition of targeting cycle problem with a master LP and POMDP sub-problems

In order to explain the interactions, we provide formal definitions for a policy, a plan, and a contingency plan. A *policy* is a mapping from states to actions for all time steps and is the output from our POMDP sub-problems. *Appendix B*, *Sections B.1* and *B.2* show how POMDP policies are developed. *Figure B-13* illustrates one time step of a policy.

A *plan* indicates a sequence of actions to take based solely upon the time step. *Figure 4-4* shows a sample plan. From this figure you can see that the action to be performed depends only upon the current time step.

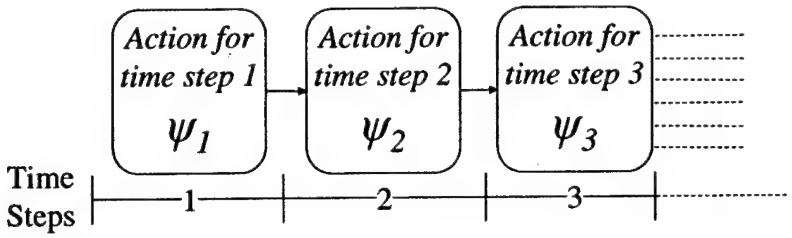

Figure 4-4: Sample Plan

An extension of this idea is the *contingency plan* that indicates a sequence of actions to take, contingent upon observations. The first step of a contingency plan contains one action and $|\Theta|$ branches which we can traverse. The second level thus contains $|\Theta|$ actions each with $|\Theta|$ branches. In our formulation, a POMDP policy is used to generate a contingency plan as described later in this chapter. *Figure 4-5* shows a contingency plan for a contact with three possible types: Not a target, Tank, and SSM.

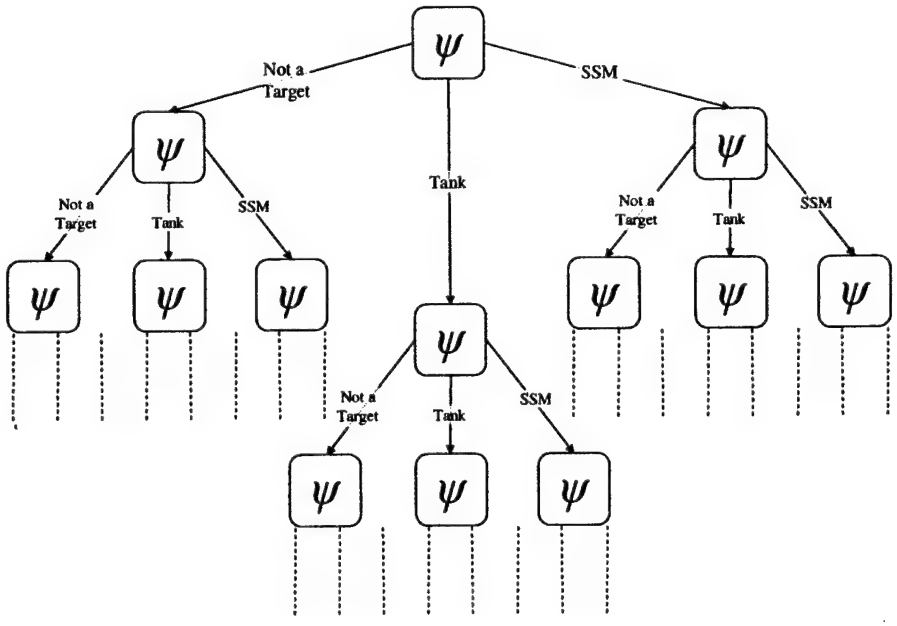

Figure 4-5: Sample contingency plan for a contact with three possible types: Not a target, Tank, and SSM.

4.5.1 Solving the LP

There is a large body of literature concerning the solution of linear programming problems. A variety of algorithms have been proposed and extensively analyzed. Polynomial time algorithms have been found along with many other important theoretical results [8]. Two main bodies of solution algorithms can be identified. The first of these groups is the set of algorithms based upon the simplex method. The simplex method, originally proposed by George Dantzig in 1947, traverses the extreme points of the feasible space as defined by the constraints. It moves from one extreme point to the next until it finds the optimal solution. Extensions and modifications have been made to Dantzig's original algorithm to deal with degeneracy and other computational issues.

Interior point methods are the other large class of algorithms that are used to solve linear programming problems. Algorithms of this type include the log-barrier, Newton's, affine scaling, primal path following, and primal-dual path following. Algorithms of this type, as the title indicates, start at a point interior to the feasible region. They then move in the feasible region toward the optimal solution.

While interior point algorithms have superior theoretic properties, the simplex method and its subsequent modifications perform quite well in practice, so we consider another criterion in picking which type of algorithm to use. In solving the targeting cycle problem, we are not solving a single LP. Rather, we are solving a large number of similar LPs that vary only in that later LPs have a superset of variables with respect to earlier LPs. In considering this note that, upon termination, the simplex method is at the optimal point for the original problem which would in turn be a feasible solution to a subsequent LP. In fact, the point at which the simplex method terminates might be relatively close to the optimal solution for the new problem. Rather than starting over, we can begin the simplex method at the previous optimal basis and take advantage of the computation performed for the previous problem.

This is not possible when interior point methods are used. The power of interior point methods is their ability to traverse the feasible region freely, without a large hindrance from the constraint set. Thus, starting an interior point method from the previous optimal solution would reduce it to an algorithm much like the simplex that moves along the exterior portion of the feasible region. Each iteration of an interior point method, however, requires more work than an iteration of the simplex method. Rather than starting at the previous optimal solution, we can

start an interior point method at another interior point. This provides no guarantee that we are close to the optimal solution so we cannot make use of the computational effort expended in the previous iteration.

For these reasons and due to the widespread availability of optimization packages that implement the simplex method, we use it to solve the master LP. Full details of the simplex algorithm can be found in [8].

4.5.2 Solving the POMDPs

Of the POMDP solution algorithms discussed in *Chapter 3*, two were selected, Incremental Pruning and Linear Support. The Linear Support algorithm, with its ability to approximately solve POMDPs, would be the sole choice if it were not for the complexity of finding extreme points in higher dimensional spaces. For the two-state target POMDPs, the extreme point enumeration step described in *Appendix B, Section B.1.1*, is trivial. When the size of the state space increases, this procedure becomes more difficult. This in turn makes the Linear Support algorithm more difficult. For this reason, we use Incremental Pruning to solve POMDPs for areas of interest and contacts, both of which might have more than two states.

4.5.3 Interactions

There are two types of interaction between the master LP and the POMDP sub-problems. Simplest of the two types of interactions is the passing of dual information from the master LP to the POMDP sub-problems. These duals provide the marginal value of one unit of each resource. The column of duals, $\mathbf{p}$, is found by using equation (4.25) in which $\mathbf{c_B}$ and $\mathbf{B}$ correspond to the objective function coefficients and columns, respectively, for the variables in the optimal solution.

$$\mathbf{p} = \mathbf{c}_\mathbf{B}'\mathbf{B}^{-1}. \tag{4.25}$$

From $\mathbf{p}$ we get the dual values from equations (4.2) to (4.6) that we need to pass to the POMDP sub-problems for the calculation of C_{ik}^{ψ}. In actual implementation, the dual values are retrieved from the LP solution package through a function call. Once these duals are received and C_{ik}^{ψ} calculated for all objects, actions, and epochs, the POMDP sub-problems can be solved, yielding optimal policies which are then used to generate improving contingency plans.

This process is not as straightforward as it might seem. A POMDP solution algorithm such as Linear Support or Incremental Pruning does not generate a contingency plan. Rather, it generates a policy mapping belief states to actions. From this policy, we must extract the contingency plan.

Note that, given an initial belief state, we can only visit a finite number of belief states over a finite horizon. For example, if we begin with a target that is fully alive and we perform action ψ on that target, we will receive one of two observations, live or dead. Each of these action-observation pairings has an associated belief state; thus, the number of belief states we must consider in time step two is two, three is four, four is eight, and so forth. The branching factor of this tree is the number of possible observations, so these trees can become quite large for long horizons or for contacts with a large number of possible observations.

To explain how contingency plans are generated, we must first introduce the idea of a *belief state node*. A belief state node contains the observation for which the node was generated, a belief state, the optimal action associated with that belief state at that epoch, and the probability of arriving in that belief state. A contingency plan is built by generating the belief state nodes for all the belief states we may transition to over the planning horizon. *Figure 4-6* shows the belief state node representation of a contingency plan for a target.

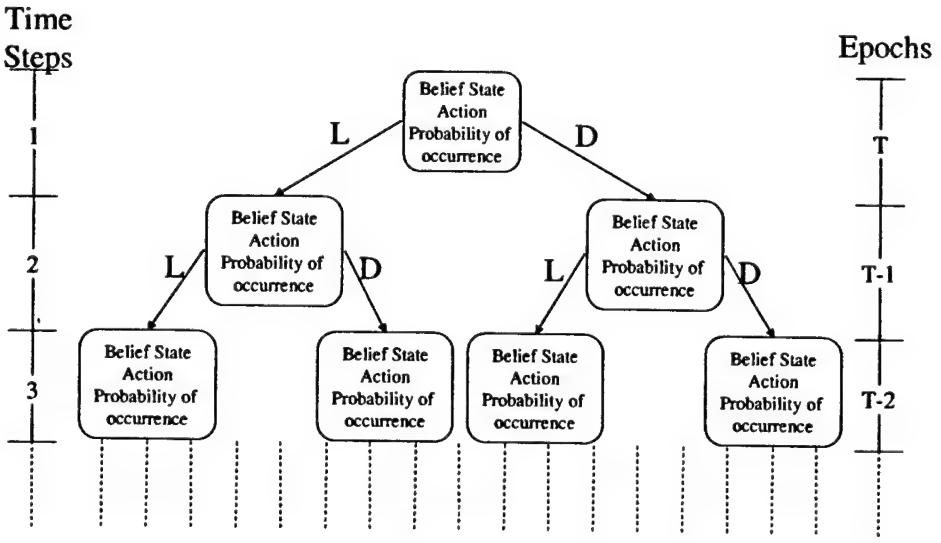

Figure 4-6: Contingency Plan for a target with first three time steps shown. L and D represent live and dead observations.

Using the Baysean update equation, equation (3.7), with the associated belief state and action from the parent node as well as the observation for which the child node was generated, we can calculate the child node's belief state. To determine the action associated with the node, we simply do a table lookup to find the alpha vector, at the appropriate epoch, that dominates at the node's belief state.

To calculate the probability of occurrence, $\mathcal{Y}_{\pi}^{\psi\theta}$, for a belief state node with associated observation θ and parent node with a belief state of π, action ψ, and probability of occurrence $\mathcal{Y}_{\mathrm{P}}$ we use (4.26).

$$\mathcal{Y}_{\pi}^{\psi\theta} = \mathcal{Y}_{\mathrm{P}} \sum_{s' \in S} \sum_{s \in S} \pi(s) \mathcal{E}_{ss'}^{\psi} \mathcal{O}_{s'}^{\psi\theta} . \tag{4.26}$$

When the entire contingency plan has been generated, we calculate the necessary data to add a column to the master LP. There are two parts to this data: expected reward and expected resource usage. Calculating the expected resource usage for use in equations (4.2) to (4.5) follows the same procedure for all of the POMDP types. For each belief state node, the resources necessary (e.g. aircraft, weapons, etc.) to carry out the associated action are multiplied by the probability of occurrence, $\mathcal{Y}_{\pi}^{\psi\theta}$. These values are then summed across belief state nodes at the same level for aircraft and sensor usage and across all belief state nodes for weapons usage and aircraft attrition. This provides the constraint data for equations (4.2), $\mathrm{NA}_{\mathrm{aoit}}$, (4.3), $\mathrm{LKS}_{\mathrm{boit}}$, (4.4), $\mathrm{WE}_{\mathrm{woi}}$, and (4.5), $\mathrm{ATTA}_{\mathrm{aoi}}$. Generation of the objective function value, however, depends upon the type of POMDP we are solving.

The objective function value for an area of interest contingency plan is a function of the expected number of discoveries in that area. To calculate this quantity, we store an additional piece of data in belief state nodes that correspond to an area of interest POMDP. This extra piece of data is the expected number of discoveries when the optimal action is performed at the current belief state. This number is multiplied by $\mathcal{Y}_{\pi}^{\psi\theta}$ and then summed for all the belief state nodes in the contingency plan yielding $\mathrm{EDIS}_{\mathrm{oi}}$. The belief state for an area of interest, a PMF over the number of objects in the area, is not directly used because of possibility of movement between areas, which, though not considered here, is an important extension.

The objective function value for a contact contingency plan depends upon the probability of declaring a contact as a certain type. This is found for each element of Ξ_i by summing up

$\mathcal{Y}_{\pi}^{\psi\theta}$ for the nodes with "Declare" actions for the appropriate contact possible type. This summation yields $PDEC_{\xi oi}$ for all $\xi \in \Xi_i$.

Columns generated by target POMDPs have objective function coefficients based upon the belief that the target is dead by the conclusion of the contingency plan. This is calculated by updating the belief state, which for targets is the belief that the target is dead, for all nodes in the lowest level of the tree using the optimal action. Each new belief state is multiplied by the probability of occurrence for its associated node, $\mathcal{Y}_{\pi}^{\psi\theta}$. These values are summed, resulting in the probability the target is dead at the end of the contingency plan. We do not consider observations received after the last step in the planning horizon because we are unable to take advantage of the information they provide.

This brings up an interesting characteristic of area of interest contingency plans. While the contact and target contingency plans branch upon the observations received, area of interest contingency plans do not have this characteristic. This is due to our belief state representation. For contacts, our belief state is our belief that the contact is of a certain type and as stated earlier, our belief state for targets is the belief that the target is dead. Observations that we receive help update this information. Our belief state for areas of interest is a PMF over the number of objects in that area. When a number of objects are discovered, we update our belief state using only the transition probabilities, $\mathcal{E}_{ss'}^{\psi}$. We do not use this information as a separate observation about the state of the system. Inferences based upon the number of objects discovered would be problem specific and difficult to quantify for a number of reasons. An example of using the number detected as an observation would be increasing the probabilities of the higher states when a large number of objects are discovered. The logic behind such an inference is that military resources are usually grouped together for mutual defense. Adding such observation characteristics to area of interest POMDPs could be done but in addition to the logic governing the updates, the finite size of the belief state PMF must be taken into account. *Figure 4-7* illustrates a contingency plan for an area of interest with the appropriate information in the belief state nodes. Note that this contingency plan has collapsed into a plan in which the optimal action is based solely upon the time step.

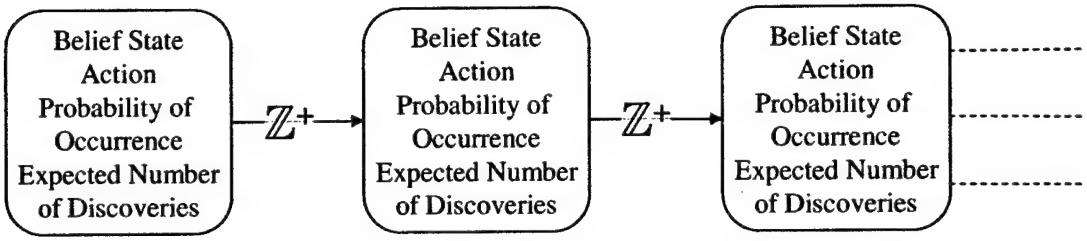

Figure 4-7: Contingency plan for an area of interest. Note that we do not branch upon observations.

4.6 Chapter Summary

Using the hybrid framework originally proposed by Yost, we see that the entire targeting cycle problem can be solved using the LP/POMDP formulation. Our addition of areas of interest, contacts and regenerative targets as objects to act upon as well as a new initialization and new integer formulations provide a broader framework for solving the targeting cycle problem. The intricacies of implementing this algorithm, along with some specific assumptions and problem characteristics, yield a variety of different tests that can be run to determine the technical and real-world behavior of the algorithm.

[This Page Intentionally Left Blank]

5 Scenarios, Results, and Analysis

While the theoretical properties of the LP/POMDP decomposition and solution of the targeting cycle problem are important, they do little to provide information about how well the approach could provide real-time assistance to military planners. We know that the POMDP sub-problems are PSPACE-complete [33], see [36] for a brief overview of complexity theory. Essentially, a problem being PSPACE-complete means that it is in the set of hardest problems in PSPACE where PSPACE is "the set of decision problems that can be solved by a Turing machine using a polynomial amount of memory, and unlimited time" [36].

5.1 Testing the Algorithm

Yost proved the theoretical validity of the LP/POMDP decomposition and presented preliminary solution analysis. We delve further into our solutions, specifically, the selected contingency plans and the characteristics of these contingency plans under a variety of conditions. We divide these experiments into two groups, those that test computational characteristics of the formulation and those that demonstrate the formulation's use in a variety of planning scenarios. Each experiment is set up with a specific hypothesis in mind and a scenario is constructed to test that hypothesis. We then take the results from each scenario and draw conclusions about the hypothesis. To perform these tests we begin with a basic scenario and then vary input parameters and data.

5.1.1 Basic Scenario

The basic scenario we consider closely mirrors current and projected Air Force operations. Aircraft are divided into five groups: strike aircraft with large capacity, strike aircraft with small capacity, ISR aircraft with a wide variety of sensors, ISR aircraft with a more limited set of sensors, and composite aircraft that combine some characteristics of strike and ISR aircraft. In keeping with the current Air Force vision, three types of munitions are employed. The Joint Direct Attack Munitions (GBU 31/32) are GPS guided munitions for destroying hardened targets and vehicles. AGM-65 Maverick missiles are guided by infrared, electro-optical, or thermal sensors and extremely versatile. However, they do not perform well against hardened buildings or troop deployments. Finally, we consider the use of cluster munitions, such as the CBU-52B mentioned in *Chapter 2*, used against lightly armored vehicles or personnel. Enemy targets include C2 facilities, surface-to-surface missiles (SSM), early warning (EW) radars, tanks, and other military equipment. *Table 5-1* summarizes the aircraft, weapons, and targets considered in the targeting cycle problem basic scenario.

Aircraft	Weapons	Targets
Large Sensor	Seeker Missile	C2 Facility
Small Sensor	GPS Bomb	SSM
Large Weapon	Cluster Bomb	Tank
Small Weapon		Supply Truck
Small Combo		Mobile HQ
		EW Radar

Table 5-1: Summary of aircraft, weapons, and targets used in the targeting cycle problem basic scenario.

Figure 5-1 spells out the characteristics of the basic problem, which will be used as a starting point for all of the structural variations and targeting cycle vignettes. Dual initialization, φ- control, action control, and the first-action-same MIP formulation are those described in *Chapter 4*.

<table>
<tr><th colspan="4">Basic Scenario</th></tr>
<tr><td rowspan="2">Targets—Value
2 C2 Facilities---5
2 SSMs---50
2 Tanks---5
2 Supply Trucks---3
2 Mobile HQ---5
2 EW Radars---5</td><td colspan="2" rowspan="2">Actions
Large Sensor
Small Sensor
Small Combo
Large Weapon + Seeker Missile
Large Weapon + GPS Bomb
Large Weapon + Cluster Bomb
Small Weapon + Seeker Missile
Small Weapon + GPS Bomb
Small Weapon + Cluster Bomb
Small Combo + Seeker Missile
Small Combo + GPS Bomb
Small Combo + Cluster Bomb
Large Weapon + 2 Seeker Missiles
Large Weapon + 2 GPS Bombs
Large Weapon + 2 Cluster Bombs
Small Weapon + 2 Seeker Missiles
Small Weapon + 2 GPS Bombs
Small Weapon + 2 Cluster Bombs</td><td>Aircraft
1 Large Sensor
1 Large Weapon
2 Small Sensor
2 Small Weapon
2 Small Combos</td></tr>
<tr><td rowspan="2">Allowable Attrition
1 Large Sensor
1 Large Weapon
1 Small Sensor
1 Small Weapon
1 Small Combo</td></tr>
<tr><td>Munitions
10 Seeker Missiles
5 GPS Bombs
10 Cluster Bombs</td><td colspan="2">φ-Control for POMDPs</td><td>Action Control for POMDPs with update interval of 10</td></tr>
<tr><td>Horizon
8 periods at 30 minutes per period</td><td colspan="2">First Action Same MIP</td><td>Dual Initialization</td></tr>
<tr><td colspan="4">Sensors coupled with aircraft thus no constraints beyond aircraft usage</td></tr>
</table>

Figure 5-1: Basic Scenario

Due to the large number of combination actions, in which an ISR platform is paired with a strike platform, e.g. an action that uses a Large Weapon aircraft with a Seeker Missile together with a Large Sensor aircraft, we do not list them in *Figure 5-1*. However, such combination actions are used in the formulation. ISR aircraft can jam the radar of antiaircraft systems thus reducing their effectiveness and the probability of attrition. Pairing strike and ISR aircraft can also provide immediate BDA.

5.1.2 Structural Variations

In testing the structural characteristics of the LP/POMDP formulation and algorithm, we consider five main areas: varying values of the allowable POMDP value function error (φ), the update interval for action control, short versus long planning horizons (T), initialization techniques, and IP/MIP formulation.

- **Impact of POMDP Value Function Error Tolerance**

  ➢ **Preset POMDP Error**

Hypothesis: Increase in POMDP error tolerance only slightly degrades solution but greatly reduces solution times. Refer to *Chapter 3*, *Section 3.3.2.1* and *Appendix B*, *Section B.1.2* for discussion about the POMDP value function error tolerance φ.

Scenario: Basic scenario with POMDP error tolerances of 0, 0.00001, 0.0001, 0.001, 0.01, 0.1, and 1.

Conclusion: As hypothesized, the solution times were markedly improved when looser POMDP value function error tolerances were used and there was little impact upon the optimal LP objective function value. See *Section 5.4.1* for the computational results.

➢ **φ-Control**

Hypothesis: The LP/POMDP formulation can be solved with progressively smaller POMDP value function error tolerances with no affect upon the optimal solution but with greatly reduced solution times. Refer to *Chapter 4, Section 4.4.5* for a discussion of *φ-control.*

Scenario: Basic scenario with initial POMDP error tolerance of 1. When the stopping criteria described in *Chapter 4, Section 4.2* are met, φ is reduced by a factor of 10 and the algorithm continued. This is done until $\varphi \leq \varepsilon$, the numerical error tolerance used in this work.

Conclusion: As hypothesized, the solution times were considerably improved when *φ-control* was used and there was no impact upon the optimal LP objective function value. See *Section 5.4.1* for the computational results.

- **Action Control with Varying Update Intervals**

Hypothesis: Different update intervals will speed up the algorithm due to the reduced number of actions considered by each POMDP. Refer to *Chapter 4, Section 4.4.5* for a discussion of action control.

Scenario: Basic scenario with action control off and action control on with update intervals of 2, 5, 10, and 15.

Conclusion: As hypothesized, the solution times were shorter for some action control update intervals and but were longer for others. There was no impact upon the optimal LP objective function value. See *Section 5.4.2* for the computational results.

- **Variations in Planning Horizon**

Hypothesis: Longer planning horizons will increase the solution times but will allow for improved objective function values due to the increased number of time steps that an object can be acted upon.

Scenario: Basic scenario with horizons from 4 to 15.

Conclusion: As hypothesized, solution times for shorter horizons were less than those for longer horizons. However, the change in the optimal LP objective function value that we expected was not present. See *Section 5.4.3* for the computational results.

- **Plans versus Dual Initialization Techniques**

Hypothesis: Dual initialization will produce the same optimal solution in fewer iterations and less time. Refer to *Chapter 4*, *Sections 4.2* and *4.3* for descriptions of plans and dual initialization.

Scenario: Basic scenario with plans and dual initialization.

Conclusion: As hypothesized, dual initialization yielded the same optimal LP objective function value as policy initialization, in a shorter period of time and less iterations. See *Section 5.4.4* for the computational results.

- **Different IP/MIP Formulations**

Hypothesis: Varying MIP formulations have little impact upon the optimal value but have definite impact upon the solution time. Refer to *Chapter 4*, *Sections 4.2* and *4.4.4* for a full explanation of the three IP/MIP formulations.

Scenario: Basic scenario with all binary variables, contingency plans with first action pause mixed, and all contingency plans with same first action mixed.

Conclusion: As hypothesized, the values attained for the three IP/MIP formulations were almost identical but the all binary and first action pause formulations did not solve in a reasonable amount of time and thus were stopped after 500 seconds. See *Section 5.4.5* for the computational results.

5.1.3 Targeting Cycle Vignettes

While the structural vignettes aim to demonstrate the characteristics of the algorithm based upon adjustable parameters that are independent of the application, these targeting cycle vignettes probe the behavior of the algorithm when characteristics of the planning scenario are changed.

- **Basic Scenario**

This scenario serves as a baseline from which to compare the later vignettes. Its characteristics are listed in *Figure 5-1*. Computational results for all of the vignettes are listed in *Section 5.5*. See *Section 5.5.1* for a full discussion of this vignette's solution.

- **Regenerative Targets**

Hypothesis: Regenerative targets will be examined more often and struck more often.

Scenario: Basic scenario with C2 facilities regenerating at different rates.

Conclusion: Contrary to our hypothesis, regenerative targets were not always examined and struck more often. If the target began in the live state, it was more efficient to wait to act upon the target until the final time steps. If, however, the target began in a belief state closer to 0.5, then it was examined and struck more often. See *Section 5.5.2* for a full discussion of this vignette's solution.

- **Antiaircraft Threats**

Hypothesis: Increased rates of attrition due to antiaircraft threats will induce strategies that are more conservative and the use of jamming.

Scenario: Basic scenario with addition of 2 medium and 2 long SAMs covering all other targets.

Conclusion: Due to the relative availability of jamming resources, there was little change from the basic scenario. See *Section 5.5.3* for a full discussion of this vignette's solution.

- **Object Discovery**

Hypothesis: Benefit to be gained from object discoveries drives the use of ISR and combination resources in object discovery actions.

Scenario: Basic scenario with addition of 2 areas of interest with varying PMFs over the number of objects in the area and average value of objects in the area.

Conclusion: As hypothesized, ISR resources were used for object discovery thus increasing the optimal LP objective function value. See *Section 5.5.4* for a full discussion of this vignette's solution.

- **Contact Identification**

Hypothesis: Benefit to be gained from contact identification will cause a shift of ISR and combination resources to identification actions.

Scenario: Basic scenario with addition of 3 contacts with varying PMFs over object types.

Conclusion: As hypothesized, ISR resources were used for contact identification thus increasing the optimal LP objective function value. See *Section 5.5.5* for a full discussion of this vignette's solution.

- **Complete Targeting Cycle**

Hypothesis: Combining all portions of the targeting cycle will result in a balanced strategy that deals with areas of interest, contacts, targets, both regenerative and nonregenerative, and the potential for attrition.

Scenario: Basic scenario with the additions described in the previous four bullets.

Conclusion: As hypothesized, a balanced strategy was developed that dealt with the potential for attrition while still finding objects, identifying contacts, striking targets, and performing BDA. Modeling the complete targeting cycle problem provided marked improvement over solutions to the individual problems. See *Section 5.5.6* for a full discussion of this vignette's solution.

5.2 Building a Contingency Plan from a POMDP Policy

To illustrate the process described in *Chapter 4*, *Section 4.5.3*, we will use a policy generated by a target POMDP and generate three steps of the contingency plan associated with that policy given that the target starts in the live state. We begin with the value function for epoch T, which is the same as time step 1, as illustrated in *Figure 5-2*. We begin at belief state π_1, which is equal to 0 because we know the target is in the live state, and find that the optimal action is to use a Small Weapon aircraft with a GPS bomb in conjunction with a Large Sensor aircraft. This action is added to our *contingency plan* along with the associated probability of occurrence of 1 because we know the initial belief state. We then update π_1 to π_1' based upon the transition probabilities of that action.

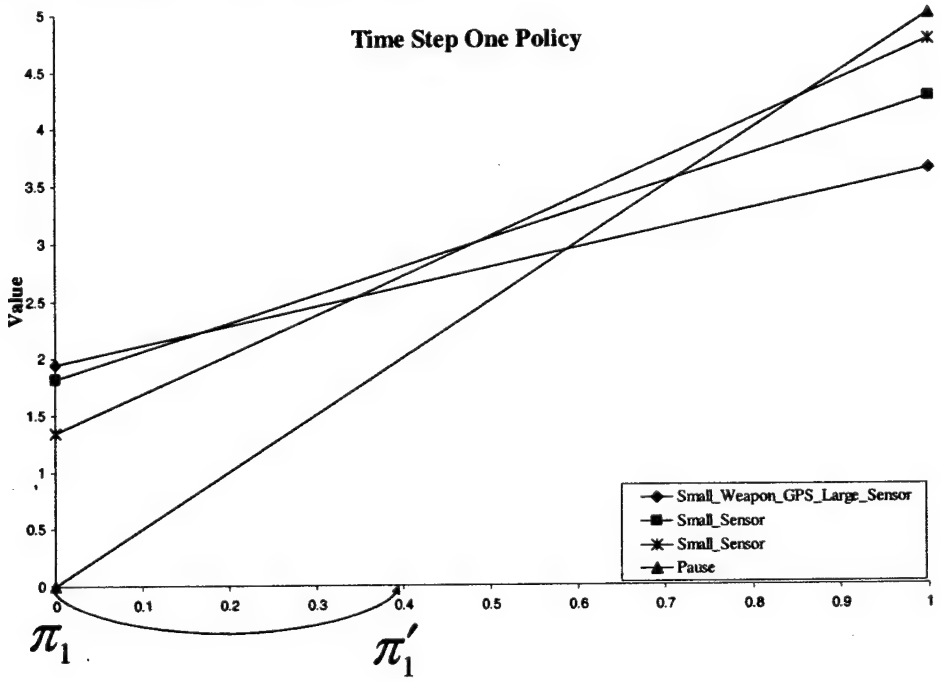

Figure 5-2: Time step one policy. We begin at π_1 and find that the optimal action at that belief state is to use the Small Weapon aircraft with a GPS bomb in conjunction with a Large Sensor aircraft. We then update π_1 to π_1' based upon the transition probabilities of that action. As a reminder, the belief state represents the probability that the target is dead.

Once we have updated our belief state to π_1' we then move to epoch T-1, which is the same as time step 2. Based upon a dead observation we would move to belief state π_{2_1} which again has an optimal action of using a Small Weapon aircraft with a GPS bomb in conjunction with a Large Sensor aircraft. If, however, we receive a live observation from the epoch T action, we would move to belief state π_{2_2} which has an optimal action of using only a Large Sensor aircraft. *Figure 5-3* illustrates these updates. Also of note in *Figure 5-3* is that while there is not much change in the shape of the optimal value function, the ISR action that dominates over the middle of the belief space has changed. These two optimal actions for epoch T-1 are added to our contingency plan along with the associated probabilities of occurrence, $\gamma_\pi^{\psi\theta}$, as defined by equation (4.26) in *Section 4.5.3*.

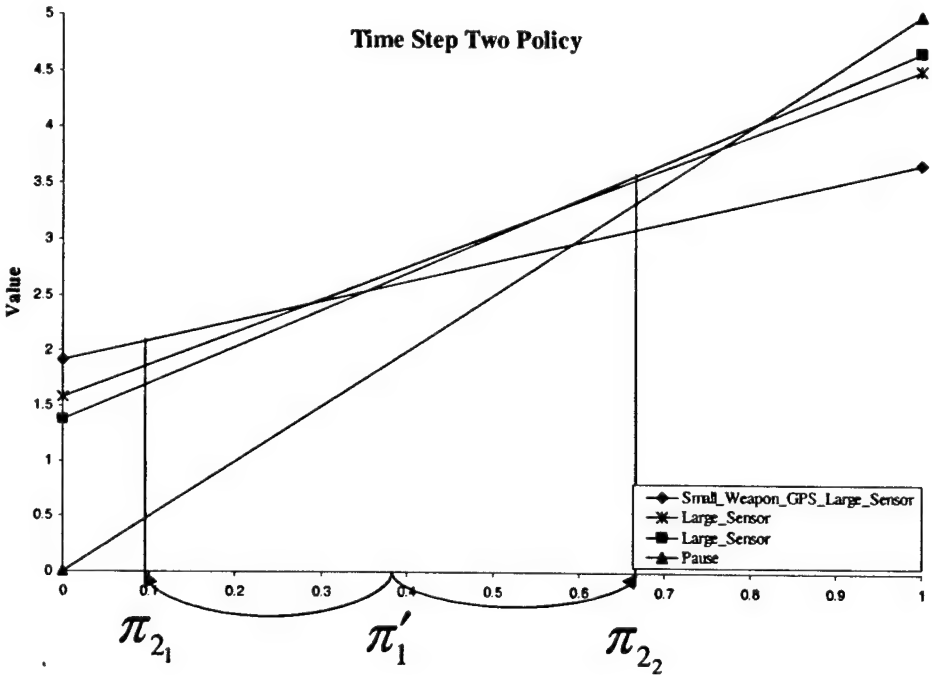

Figure 5-3: Time step two policy. Based upon the updated belief state from time step 1 we move to either π_{2_1} or π_{2_2} depending upon if we receive a live or dead observation, respectively. From these updated belief states, we find the two optimal actions for these belief states and add them to our contingency plan.

When we have found π_{2_1} and π_{2_2} we can update them to π_{2_1}' and π_{2_2}' using the transition probabilities of their respective optimal actions. *Figure 5-4* illustrates these updates.

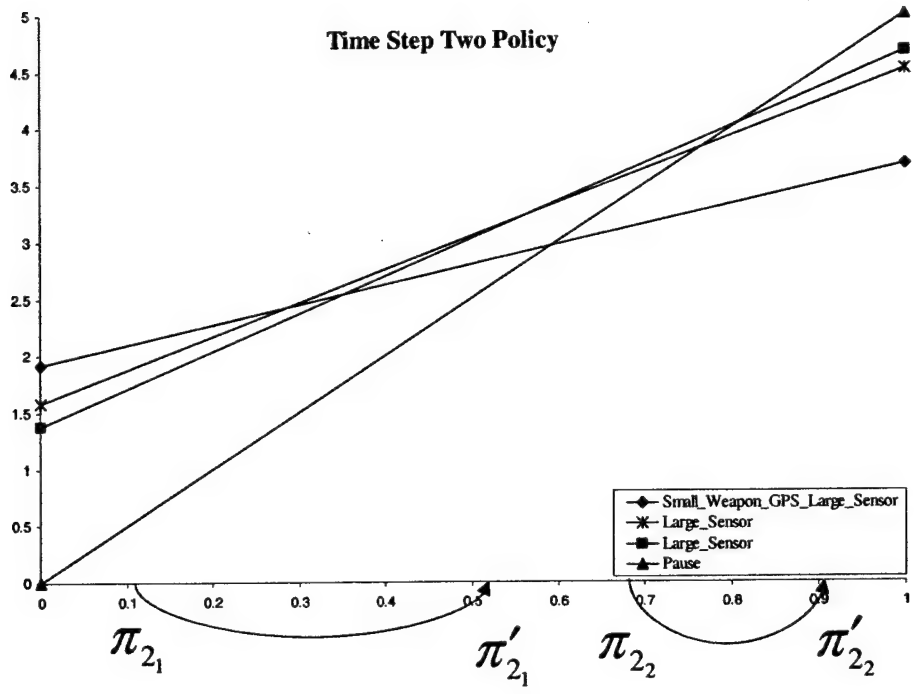

Figure 5-4: Update of π_{2_1} and π_{2_2} to π'_{2_1} and π'_{2_2} based upon their respective optimal action transition probabilities.

When we have calculated π'_{2_1} and π'_{2_2} we can move to time step 3, epoch T-2, and determine the four belief states to which we could possibly move. From π'_{2_1} we could move to π_{3_1} based upon a live observation and π_{3_2} for a dead observation. Similarly, from π'_{2_2} we could move to π_{3_3} if we receive a live observation and π_{3_4} if we receive a dead observation. *Figure 5-5* illustrates these updates as well as another change in the ISR action that dominates over the middle portion of the belief space. The optimal actions for these four belief states are added to the contingency plan along with their associated probabilities of occurrence.

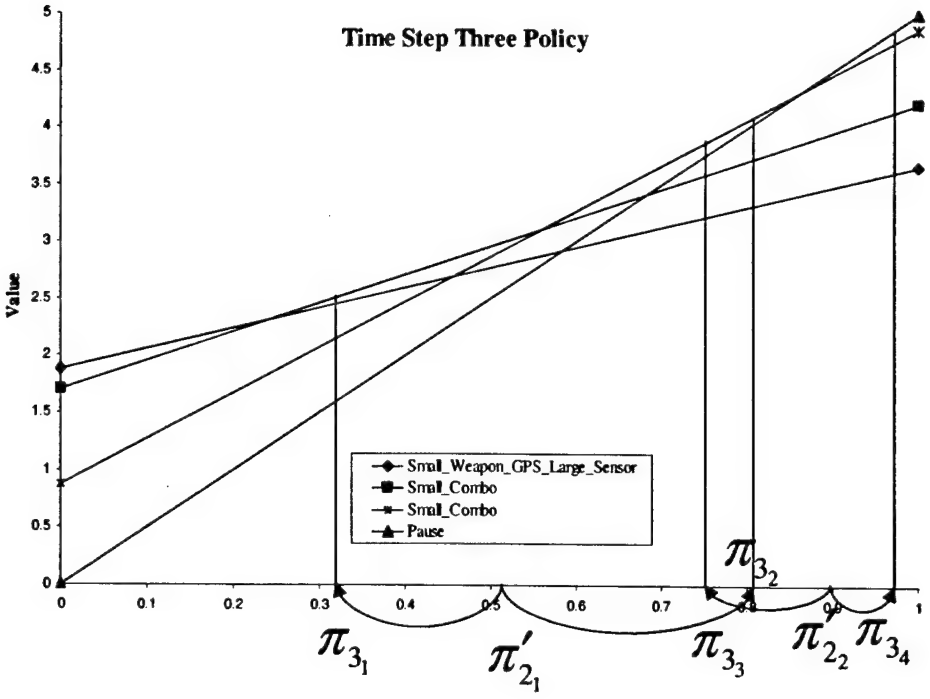

Figure 5-5: Time step three policy. Belief state π'_{2_1} would be updated to π_{3_1} based upon a live observation and π_{3_2} for a dead observation. We find the optimal actions for these belief states and add them to the contingency plan. Similarly, π'_{2_2} would be updated to π_{3_3} if we receive a live observation and π_{3_4} if we receive a dead observation. The optimal actions for these belief states are also added to the contingency plan.

Using the information gathered to this point, we can build the three step contingency plan based upon this POMDP policy, shown in *Figure 5-6*. Note the branching upon observations that occurs. Specifically, note that after two dead observations, a pause action is taken against the target but that even with two live observations, after two strikes, the contingency plan calls for an ISR action to better assess the state of the target. This illustrates the balancing between the probability of destruction and the reporting accuracy of the sensors. The value of the probability of occurrence is included in the first belief state node for illustration purposes only. Values for subsequent nodes depend upon the optimal action for the first node. Refer to *Chapter 4, Section 4.5.3* for the use of these probabilities in calculating objective function and constraint coefficients.

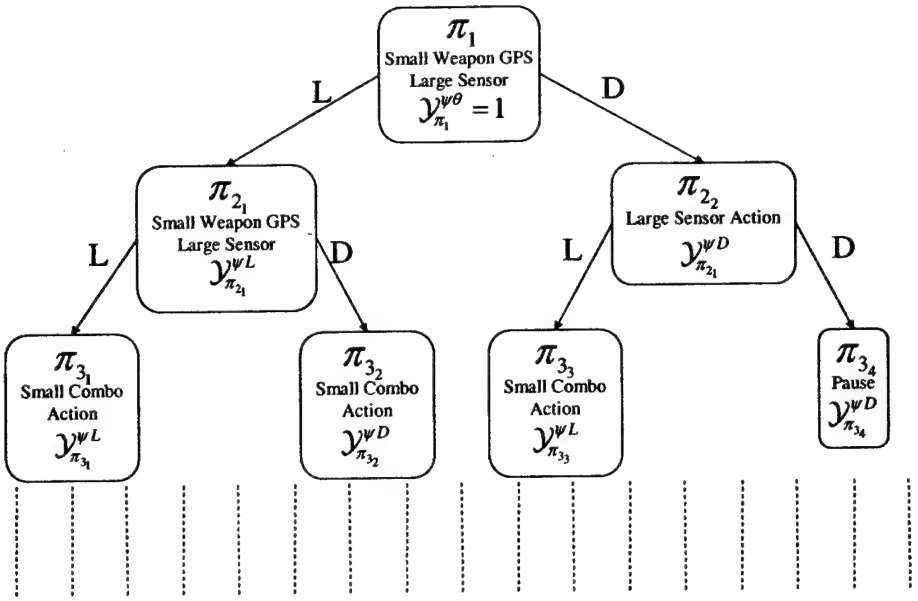

Figure 5-6: Three-step contingency plan generated from the three time steps of the POMDP shown above. Note the branching upon observations and specifically the fact that after two dead observations, a pause action is taken against the target.

5.3 Metrics

In determining the effectiveness of each variation and vignette, there are two main areas of interest: *optimal objective function value* and *solution time*. Interestingly, there is not necessarily a trade off between the two. While simply stopping the algorithm after a shorter time will decrease the optimal value attained, techniques such as *φ-control* and *action control* seek to reduce solution time while maintaining the same optimal value. In considering the optimal solution, we will look at the linear programming optimal solution, the integer programming optimal solution, and the gap between the two. The solution time for each vignette or variation is divided into four parts: the time taken to initialize the algorithm, time spent solving the master LPs, time spent solving the POMDP sub-problems, and the time spent in the branch-and-bound search for an integer solution. Also of interest is the number of iterations between the master LP and the POMDP sub-problems necessary for the algorithm to converge, which provides insight into how quickly the algorithm will converge, independent of the speed of the computer it is being run on.

5.4 Structural Variations

The algorithms to solve the targeting cycle problem were implemented in Java on a computer with 384MB of RAM and a 1 GHZ Pentium III processor. Linear and integer programming problems were solved with XPRESS 2003C.

Table 5-2 shows the preferred trends for the metrics we will be considering. We would like the LP and MIP optimal solutions to be as high as possible while we would like the gap between the LP and integer solutions, all parts of the solution time, and the number of iterations to be as low as possible.

	Optimal Solution			Solution Times (sec.)					Number of Iterations
Variation	LP	MIP	Gap	Total	Initialize	LP	POMDP	MIP	
	↑	↑	↓	↓	↓	↓	↓	↓	↓

Table 5-2: Preferred trends for metrics considered in targeting cycle problem structural variations and targeting cycle vignettes.

5.4.1 Allowable POMDP Value Function Error

Table 5-3 shows the metrics for various values of φ. The φ-control variation was implemented as discussed in *Section 5.1.2*. As expected, larger values of φ allow for much faster solutions with only minor degradation of the optimal solution. It is interesting to note the difference between objective function values for the first three cases. While the LP solutions are the same to the thousands place, the MIP solution for φ equal to 0.0001 is actually higher than for 0.00001 and 0.0000001. At the same time, however, the time necessary to achieve this solution drops by a factor of ten. Even when φ is set to 1, the optimal solution is less than 1.5 percent away from the optimal solution for the most restrictive case.

	Optimal Solution			Solution Times (sec.)					Number of Iterations
Variation	LP	MIP	Gap	Total	Initialize	LP	POMDP	MIP	
φ=ε	143.7388	143.7384	4.0429E-4	5364.215	0.22	4.824	5344.368	9.213	107
φ=0.00001	143.7388	143.7385	3.0741E-4	1364.346	0.28	4.899	1349.695	4.861	109
φ=0.0001	143.7388	143.7386	1.9803E-4	520.837	0.24	4.962	507.412	4.287	108
φ=0.001	143.7387	143.7375	0.0013	166.457	0.27	4.036	153.897	3.896	102
φ=0.01	143.7372	143.7339	0.0033	62.93	0.18	3.708	46.685	6.599	96
φ=0.1	143.6047	143.5888	0.016	28.581	0.20	1.984	15.171	6.83	74
φ=1	141.9067	141.8638	0.0429	11.136	0.23	0.28	5.628	2.404	40
φ control	143.7388	143.7386	2.5468E-4	514.462	0.35	9.461	494.857	4.777	172

Table 5-3: Metrics for different values of the POMDP error, φ, as well as φ-control. Note the decrease in the optimal solutions with looser tolerances but the much smaller solution times for the POMDPs.

As expected, φ-control greatly reduced the solution time necessary for the optimal solution. In this case, successive values of φ were reduced by a factor of ten with only a slight

increase in the LP solution time due to the higher number of iterations that result from φ-control.

Figure 5-7 graphically shows the effect of φ upon the solution time. The graph is done with logarithmic scales thus the linear relationship between solution time and the POMDP error tolerance, φ, indicate an exponential growth in solution time as φ is reduced. Due to the reduction in solution times, with negligible impact upon the optimal solutions, we can confidently use higher values of φ, or, if φ-control is used, we can set the lower bound on φ higher than ε.

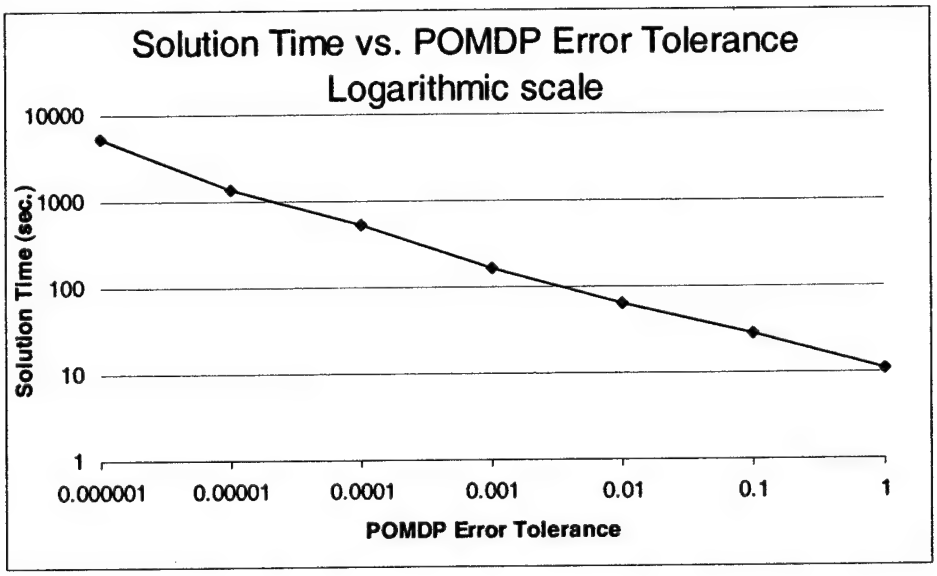

Figure 5-7: Solution time versus POMDP error tolerance on a logarithmic scale. Note that this graph is linear on a logarithmic scale thus is exponential on a normal scale.

5.4.2 Action Control Update Intervals

Table 5-4 shows the metrics for different action control update intervals as described in *Chapter 4, Section 4.4.5.* Reductions in solution times could vary depending upon how the interaction between epsilon control and action control are handled. In this implementation, we reset the actions for all objects if the stopping criteria are met but φ is not at its lower bound. The algorithm may then find actions that were not being considered that now improve the solution. Thus, action control updates occur regularly as the algorithm converges so updates might occur more often than specified. Before the algorithm terminates it is important to have φ at its lower bound and action control off. This will provide the same solution as the uncontrolled problem but will greatly reduce the time necessary to attain that solution.

As hypothesized, action control can decrease the solution time, with the lowest solution times in these tests being found with an update interval of 10.

	Optimal Solution			Solution Times (sec.)					Number of Iterations
Variation	LP	MIP	Gap	Total	Initialize	LP	POMDP	MIP	
No Action Control	143.7388	143.7386	2.0363E-4	629.948	0.321	12.867	596.766	18.781	181
2	143.7388	143.7379	8.9268E-4	581.137	0.25	8.935	541.624	29.237	159
5	143.7388	143.7371	0.0017	682.113	0.699	11.944	657.515	8.96	184
10	143.7388	143.7386	2.5468E-4	502.22	0.22	9.419	485.972	5.558	172
15	143.7388	143.7375	0.0013	769.406	0.291	10.581	748.126	9.075	186

Table 5-4: Metrics for different action control update intervals.

5.4.3 Variations in Planning Horizon

Table 5-5 shows the output from varied planning horizons. As originally thought, the solution times decreased significantly with shorter planning horizons. However, this does not affect the optimal solution as much as previously thought. Rather, the algorithm assigns most of the resources in the small number of times steps over which it has to plan rather than over a longer period of time. In the cases of T between 4 and 6, reusable assets such as aircraft and sensors were the most valuable. For T between 9 and 11, non-reusable assets such as weapons and aircraft attrition were tightly constrained. In the intermediate cases, the reusable and non-reusable assets were more equally used. The final four test cases, planning horizons between 12 and 15, did not converge after three hours of run time so they were terminated. Even with horizons of 9, 10, and 11, with φ*-control* and *action control*, the solution times are prohibitive.

While solving the problem over a sufficiently long horizon provides robustness against future events, too long of a horizon reduces the viability of such a formulation for use in real time. As computing power increases, longer horizons could be considered with the exact number of time steps being a judgment call for the operator.

Figure 5-8 shows the relationship between planning horizon and solution time. While there is a definite increase in solution time as the planning horizon increases, the exponential growth that was in *Figure 5-7* is not present.

	Optimal Solution			Solution Times (sec.)					Number of Iterations
Variation	LP	MIP	Gap	Total	Initialize	LP	POMDP	MIP	
T=4	142.2072	142.1941	0.0131	25.849	0.34	2.795	18.116	1.803	111
T=5	142.7058	142.7046	0.0012	52.047	0.23	2.953	44.538	2.724	125
T=6	143.1009	143.1009	6.8979E-6	94.008	0.25	4.039	85.893	2.022	132
T=7	143.445	143.4446	3.7593E-4	332.628	0.18	5.837	318.179	4.426	142
T=8	143.7388	143.7386	2.5468E-4	534.425	0.15	9.333	504.31	9.879	172
T=9	144.0064	144.0064	7.5608E-6	1449.206	0.26	15.683	1412.373	11.419	210
T=10	144.2359	144.2359	1.3585E-6	1293.928	0.31	27.599	1254.822	5.218	236
T=11	144.4492	144.4492	4.4846E-6	8548.453	0.3	46.642	8479.876	11.689	288

Table 5-5: Metrics for different planning horizons, T. Note that as the planning horizon increases, the solution time also increases. In addition to solution time, the memory usage also increases.

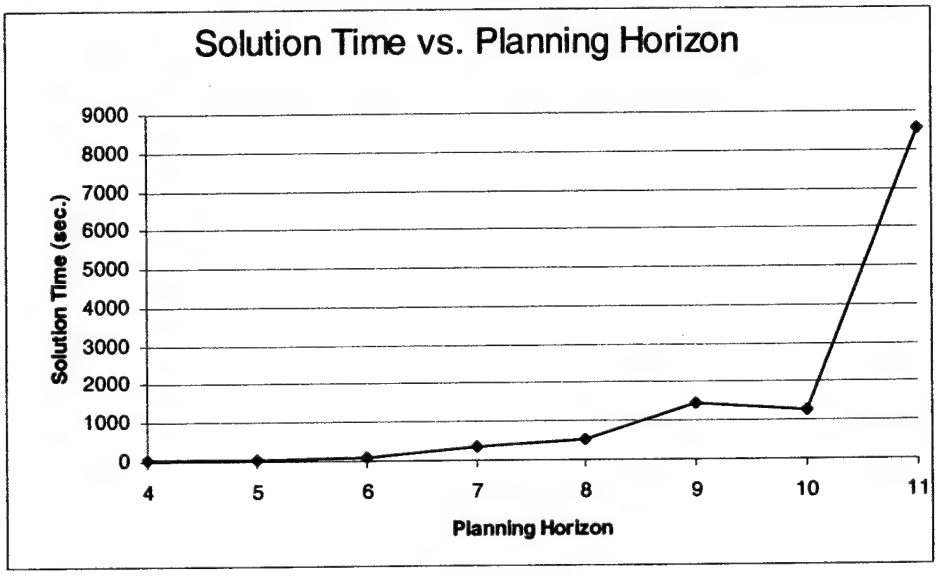

Figure 5-8: Solution time versus planning horizon. As hypothesized, the solution time increases with the planning horizon.

5.4.4 Policy versus Dual Initialization

Table 5-6 lists the metrics for policy and dual initialization using the basic scenario. While both solution techniques attain the same optimal solution value, the total solution time is quite different for the two techniques. This time difference is not found in the time spent actually initializing the algorithm. Rather, the dual initialization initiates the algorithm in such a manner that the initial dual values are closer to the final dual values. Thus, the POMDPs produce columns that are used in the optimal solution in earlier iterations. This accounts for the reduction in the number of iterations where, in the case of the LP/POMDP problem formulation, one iteration can add a sizeable amount of time. While the results below are only indicative of

one problem instance, it indicates that the dual initialization technique should be considered when implementing the LP/POMDP formulation and associated solution algorithm.

	Optimal Solution			Solution Times (sec.)					Number of Iterations
Variation	LP	MIP	Gap	Total	Initialize	LP	POMDP	MIP	
Policy	143.7388	143.738	8.676E-4	715.177	0.501	15.576	685.287	8.201	177
Dual	143.7388	143.738	2.5468E-4	523.759	0.291	9.552	501.43	6.735	172

Table 5-6: Metrics for different initialization techniques. Note that the optimal solutions are the same but policy initialization took more than 3 minutes longer to solve and required 5 additional iterations.

5.4.5 IP/MIP Formulations

As discussed in *Chapter 4*, *Section 4.4.4*, there are three different integer-programming formulations of the targeting cycle problem that can be used to find an executable solution. The first formulation forces all decision variables to be binary. The second, as proposed by Yost [37], forces those contingency plans with a first action other than pause to be binary and allows the contingency plans with pause first action to be combined as long as the total usage equals one. Our proposed formulation of allowing contingency plans to be combined, as long as their initial actions are the same, also provides an executable solution due to the rolling horizon planning framework in which we are working.

Table 5-7 lists the metrics for the three formulations. As expected, the optimal LP value, the initialization time, the LP solution time, the POMDP solution time, and the number of iterations are essentially the same for all three formulations. The total solution times are quite different for the three formulations. The branch-and-bound searches for the first and second formulations were stopped after 500 seconds. In initial trials, both were allowed to run for three hours without the optimal integer solution being found. Also of interest is the fact that the LP-IP/MIP solution gap values for the first two are still relatively small. This indicates that the algorithms could be stopped even earlier with little effect upon the solution. Rather than stopping the branch-and-bound search after 500 seconds, it could be stopped after an integer solution has been found that is sufficiently close to the LP optimal solution which is an upper bound on the MIP optimal solution. However, even if an integer solution is found immediately, the third formulation finds an optimal solution in a matter of second. Unless a problem instance dictates a different integer programming formulation, the third formulation has good performance characteristics.

Variation	Optimal Solution			Solution Times (sec.)					Number of Iterations
	LP	IP/MIP	Gap	Total	Initialize	LP	POMDP	MIP	
All Binary	143.7388	143.7345	0.0044	1012.812	0.29	9.202	491.814	510.233	172
First Action Pause	143.7388	143.7322	0.0066	999.738	0.22	9.289	481.965	507.221	172
First Action Same	143.7388	143.7386	2.5468E-4	506.484	0.281	9.307	489.575	5.919	172

Table 5-7: Metrics for different MIP formulations. Note that the optimal LP solutions are the exact same but that the MIP solutions are different. The all-binary and first action pause branch-and-bound searches were stopped after 500 seconds. The optimal integer solution was not found for either formulation within 3 hours.

5.5 Targeting Cycle Vignettes

After considering the results from the structural tests above, we implement our findings in the targeting cycle vignettes. Specifically, we use φ-control but rather than reducing φ until it has reached the numeric tolerance ε, we reduce it until it reaches 0.001. This slight modification greatly reduces solution times while having little impact upon the optimal solution generated. While the target POMDPs are solved much faster, this reduction in solution time is offset by the addition of higher dimensional area of interest and contact POMDPs that must be solved with *Incremental Pruning*, see *Chapter 4, Section 4.5.2*. *Table 5-8* shows the metrics for the six targeting cycle vignettes. Of interest are the slightly lower objective function values for the second and third vignettes. The third case is not as low as might be expected due to the relative availability of ISR assets that can reduce the effectiveness of antiaircraft threats and the relatively high allowable attrition. Note, however, that the solution times for these two vignettes increase over that of the basic scenario.

As expected, the addition of contacts and areas of interest increases the optimal objective function value due to the value gained by discovering objects and identifying contacts. Note, however, that the time necessary to solve the POMDP sub-problems greatly increases as is evident in the increased POMDP solution times present in the final three vignettes.

While the metrics shown above give a good overview of the LP/POMDP hybrid formulation, a closer look at the selected contingency plans and the resulting resource allocations provide insight into the complex interactions that the LP/POMDP formulation of the targeting cycle problem accounts for. Also, it will show how the LP/POMDP formulation improves upon the commonly used contingency plans, such as shoot-look-shoot for a target.

Variation	Optimal Solution			Solution Times (sec.)					Number of Iterations
	LP	MIP	Gap	Total	Initialize	LP	POMDP	MIP	
Basic	143.7387	143.7366	0.0021	111.607	0.261	6.522	86.645	16.806	153
Basic with Regenerative Targets	142.1012	142.1012	4.878E-5	145.163	0.3	11.275	127.037	5.078	209
Antiaircraft Threats	143.7386	143.7364	0.0022	119.272	0.261	12.74	95.885	8.844	195
Object Discovery	200.1857	200.1851	5.723E-4	1607.296	0.291	8.258	1591.805	5.209	173
Contact Identification	155.9736	155.9734	1.565E-4	888.255	0.3	7.85	873.835	5.069	167
Full Targeting Cycle Problem	229.6137	229.6137	1.768E-5	3324.087	0.491	12.763	3302.278	5.839	188

Table 5-8: Metrics for targeting cycle vignettes. While the first three vignettes have similar solution times and objective functions, the addition of contacts and areas of interest increase the solution time but also increases the objective function values.

5.5.1 Basic Scenario Solution Analysis

One manner in which to see the interactions between different targets is to see how their expected belief state changes over the planning horizon. *Figure 5-9* shows this progression for seven of the twelve targets considered in the basic scenario. To calculate the data for such a graph we define Φ_{ot} as the set of belief state nodes for contingency plan o at time step t, π_ϕ as the belief state for belief state node ϕ, and γ_ϕ as the probability of occurrence for belief state node ϕ. Thus, the expected belief state for target i at time step t, $E\pi_{it}$, is shown in (5.1).

$$E\pi_{it} = \sum_{o \in O_i} \sum_{\phi \in \Phi_{ot}} x_{oi} \pi_\phi \gamma_\phi . \qquad (5.1)$$

These $E\pi_{it}$ values are graphed in *Figure 5-9*. Note that high value targets such as the Surface-to-Surface missiles (SSM) are struck immediately with highly effective actions but that low value targets such as the supply truck are not even considered until later in the planning horizon. Also of interest are the portions of the graph such as that for the Mobile HQ 1 during time steps 2 and 3. The relative flatness of this portion of the graph indicates that the target is not being attacked. However, it is struck in time step 1 and then again in time step 4. During this pause, reusable assets such as aircraft and sensors are in use against other targets but then

become available for use against the Mobile HQ later on. Finally, note the steady progression of strikes against the C2 Facility, which is a hardened target thus requires a large number of strikes to destroy.

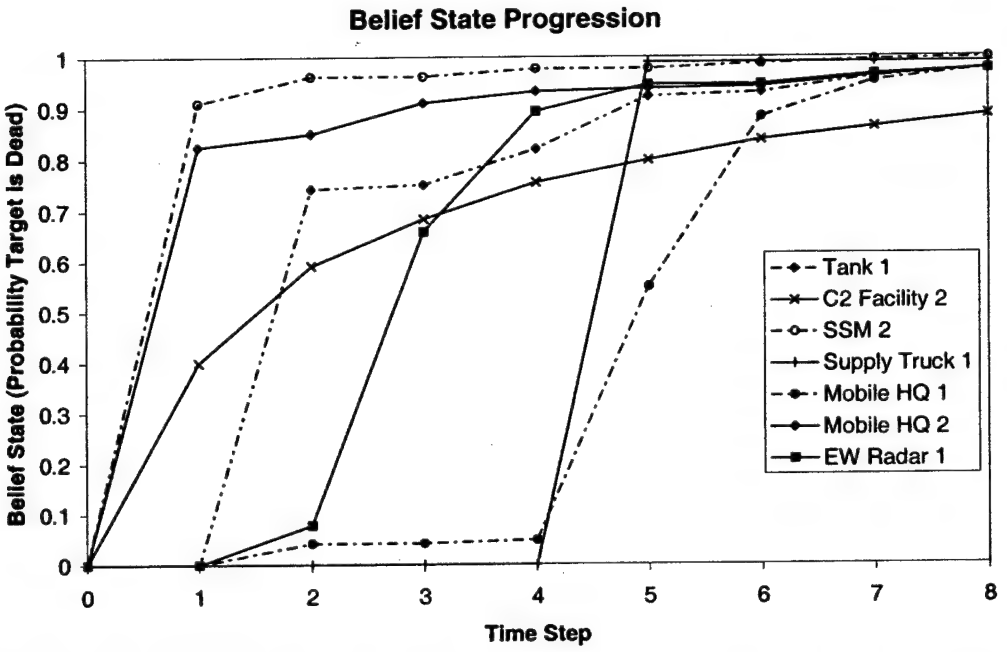

Figure 5-9: Belief state progression over the time steps for a selected contingency plan. Note that high value targets such as the SSM are struck immediately and continually checked. Also note the flat portion of the Mobile HQ 1 line in which only pause actions are taken against the target thus not changing its belief state.

To reach the belief states shown above, the master LP assigns resources to be used against each target. In the basic scenario, the most highly constrained resources were the weapons, especially the GPS bombs. The large sensor aircraft were the next most highly constrained resource.

At each iteration of the algorithm, the POMDP sub-problems provide improving columns to the master LP. The master LP considers these new columns and finds a new optimal solution. This new optimal solution might not have a higher objective function value. Rather, the master LP might shift the use of resources between different contingency plans, which changes the dual values for some or all of the resource constraints. If we were to stop the algorithm when the objective function value does not change, we might terminate the algorithm long before the optimal solution is found. This phenomenon can be seen in *Figure 5-10* which shows the objective function value for the 153 iterations necessary to solve the basic scenario. Note the horizontal sections between iterations 4 and 5 and 12 and 13. If the algorithm were terminated at

the first of these sections, we would be a full 26 percent below optimal and at the second, 13 percent.

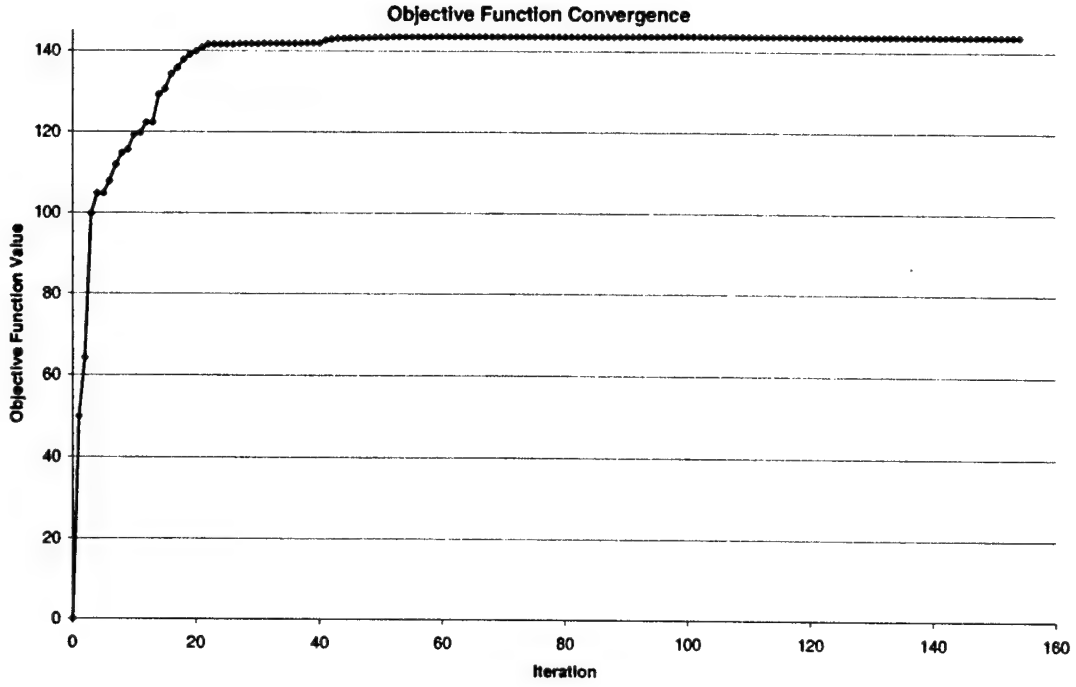

Figure 5-10: Objective function convergence of the LP for the basic scenario. Of interest are the horizontal sections between iterations 4 and 5 and 12 and 13 as well as the slow tail convergence.

Figure 5-10 also shows the slow tail convergence of the algorithm. While 99.5 percent of the optimal value is attained in the first 45 iterations, the remaining 108 iterations only provide a 0.5 percent improvement. This indicates that less restrictive stopping criteria might provide a solution that is sufficiently close to optimal in a significantly shorter time.

Contingency plans selected under the basic scenario show interesting behaviors exhibited by the LP/POMDP formulation. In addition to branching upon observations, as illustrated in *Figure 5-6*, some contingency plans act upon a target, pause for a number of time steps, and then reengage the target, as illustrated by the contingency plans represented in *Figure 5-9*. This is indicative of the balancing of resource usage between different targets, which, in the end, provides for a better solution. Such contingency plans, however, are not intuitively obvious to military planners. Rather, a human would think that we should strike a target, and continue acting upon it, be it with strike, ISR, or combination assets, until we have established that the target is destroyed. The solution to the basic scenario indicates that it is important to address a range of targets, even if specific targets are not fully transitioned to the dead state.

5.5.2 Basic Scenario with Regenerative Targets Solution Analysis

As expected, the basic scenario with regenerative targets has some of the same properties as the basic scenario. Namely, the objective function convergence shown in *Figure 5-10* is also present in this vignette. In addition, the contingency plans selected in the integer solution for nonregenerative targets have similar characteristics to those in *Section 5.5.1*. For the regenerative targets, however, there are marked differences. For one, if the regenerative target is in the live state initially, the algorithm will pause until the last few time steps and then use highly effective resources against the target. This makes sense as the target cannot regenerate if it is in the live state so using resources to move it to the dead state early on does not provide benefit. However, the interesting case arises when a regenerative target has a belief state near 0.5. In that case, the selected contingency plans continually use ISR assets to observe the target. If a live observation is received, strike resources are assigned to destroy the target. If a dead observation is received, the next action is either a pause or another ISR look. This behavior is also contingent upon the probability of regeneration for a target. Regenerative targets that regenerate with a lower probability are observed less often than those with a regeneration probability close to 0.5. If, however, a regenerative target has a probability of regeneration close to 1, it is treated much like a target that begins in the live state in that it is mostly ignored until the final few time steps when strong measures are taken to destroy the target.

5.5.3 Basic Scenario with Antiaircraft Threats Solution Analysis

As stated in *Section 5.5.1* the available weapons were the most constraining resource in the basic scenario while ISR assets were not as constraining. Thus, the addition of antiaircraft threats did not affect the optimal solution as much as we might have expected. We do observe a change in the selected contingency plans in that there are few contingency plans that use strike only actions. Rather, the combination platform, or actions that combine strike and ISR assets are used extensively. In the end, however, the result is almost the same as the basic scenario. If the number of ISR assets or the allowable attrition were more constrained, we would see more conservative strategies that do not attain as high of expected belief states as the less constrained problem.

5.5.4 Basic Scenario with Object Discovery Solution Analysis

As hypothesized, the addition of areas of interest in which objects can be discovered yields a higher optimal objective function value with the same tail convergence properties that were present in the basic scenario. *Figure 5-11* illustrates these properties.

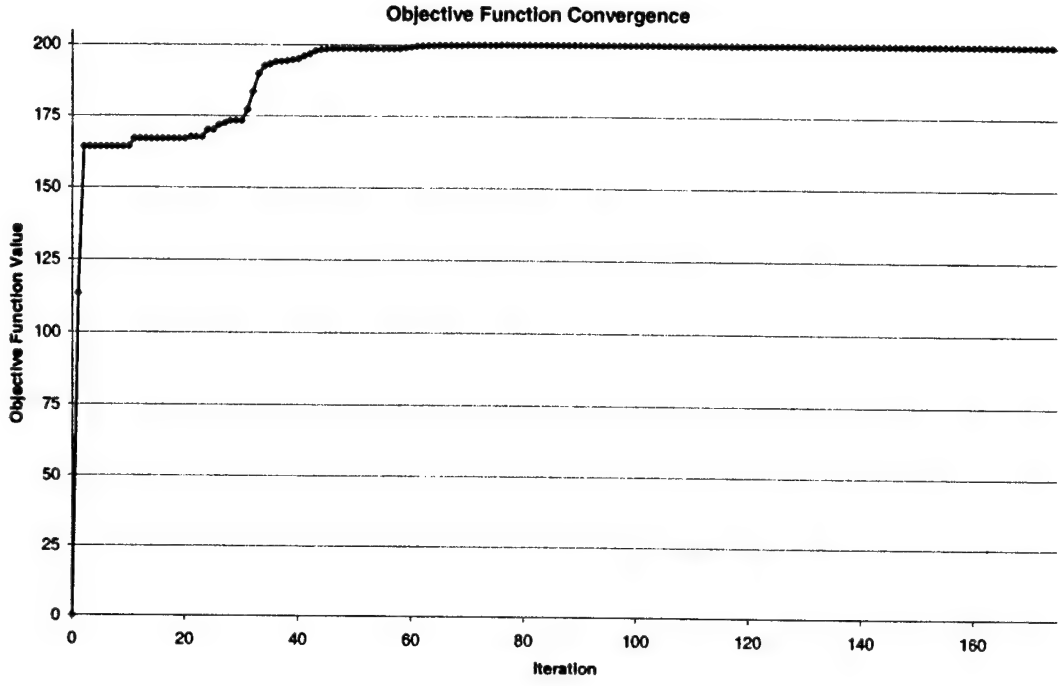

Figure 5-11: Objective function convergence of the LP for the basic scenario with object discovery. Again note the horizontal portions of the graph indicating small or no change in the objective function associated with changes in the duals. Also, we see the slow tail convergence property that was present in the basic scenario.

Also, as we had hypothesized, ISR assets are more highly constrained in this vignette than in the basic scenario. While the weapon constraints still have the highest dual values, the duals for the large sensor aircraft increase by 20 percent or more. We also see similar but less drastic gains in the small sensor and combination aircraft because of their lower discovery probabilities. The higher value of these ISR resources drives the generation of contingency plans that share them across different objects. *Figure 5-12* illustrates a contingency plan for an area of interest. Note the pauses in time steps 4 through 7. In these time steps, the ISR assets that were being used in this area of interest are being used for other tasks, such as BDA for a target or discovery of objects in another area of interest.

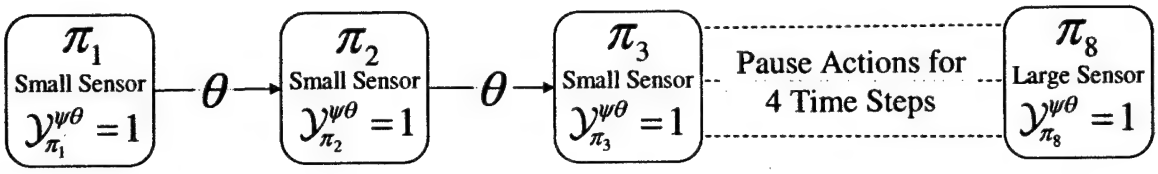

Figure 5-12: Contingency plan for an area of interest. Note the pauses in the middle where ISR assets are used elsewhere but in the last time step, the best sensor available is used.

5.5.5 Basic Scenario with Contact Identification Solution Analysis

Figure 5-13 again illustrates the properties seen in the previous four vignettes, namely slow tail convergence and flat portions of the graph indicating changes in the dual values but with little to no change in the objective function value. The increase in the objective function of approximately 12 is about what we would expect for the three contacts that were considered. Due to the large size of contact contingency plans, for the three contacts considered there were five possible target types, we will simply describe the behaviors exhibited rather than trying to graphically show them. One behavior of interest is the continued presence of pauses in the middle of contingency plans as resources are used against other objects.

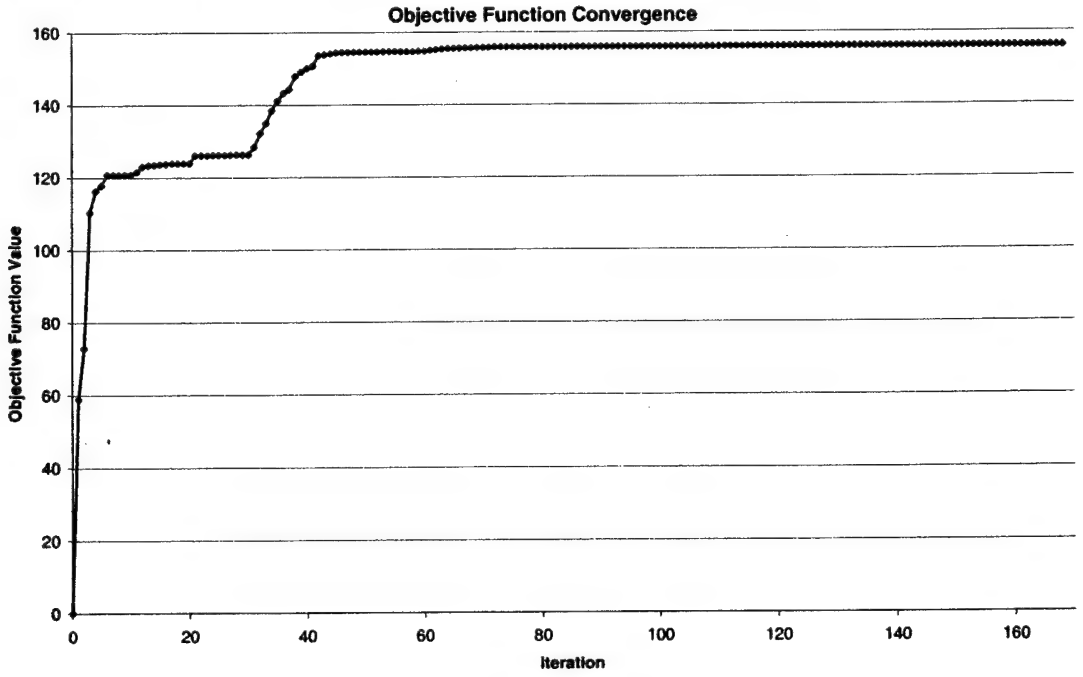

Figure 5-13: Objective function convergence of the LP for the basic scenario with contact identification. Again we see the slow tail convergence and iterations in which there is only changes in the dual values, not in the objective function.

In conjunction with this characteristic, the contingency plans that are generated and selected choose different actions for different observations at the same time step. For example, in one case when a 'Not a Target" observations is received, the contingency plan indicates a pause but when any other observation is received at that same time step, it uses an ISR asset to gather more information on the contact.

In general, as in the previous vignettes, ISR assets are more constraining than in the basic scenario. While this drives somewhat lower expected final belief states for the targets, the benefit gained by contact identification outweighs that loss. Again we see the power of the LP/POMDP formulation which carefully balances the use of resources between a myriad of tasks.

5.5.6 Full Targeting Cycle Problem

When regenerative targets, antiaircraft threats, object discovery, and contact identification are all added into the basic scenario we have a higher fidelity representation of the real-world targeting cycle problem. Solving such a problem is not trivial, however. Even with the use of action control for all of the POMDPs and φ-control for the target POMDPs, the full targeting cycle problem with 2 areas of interest, 3 contacts, and 12 targets took almost an hour to solve. As expected, the majority of this time is spent solving the POMDP sub-problems. As illustrated in *Table 5-8*, each addition to the basic problem added to the time necessary to solve the problem.

Of interest, however, is that the objective function for the full targeting cycle problem is not simply the addition of the individual contributions. The actual objective function value, 229.6, is over 8 percent higher than the sum of the individual contributions, 210.8, of which the basic scenario provides 143.7, regenerative targets reduces this by 1.6 but the addition of areas of interest and contacts add 56.4 and 12.23 respectively. This provides verification that consideration of the full targeting cycle problem rather than the individual parts yields a better solution. This improvement in the objective function value comes at a computational cost similar to the growth in the objective function value. The basic problem took 111.6 seconds to solve. The addition of regenerative targets, antiaircraft threats, object discovery, and contact identification added 33.6 seconds, 7.7 seconds, 1495.7 seconds, and 776.6 seconds respectively. This would seem to indicate that the full targeting cycle problem could be solved in 2425.2

seconds. This, however is not the case. Instead, the full targeting cycle problem took 30 percent longer, 3324.1 seconds. *Figure 5-14* again shows characteristics of the objective function value converge similar to all of the previous vignettes. Of interest is the definite staircase effect that is present in this solution. Five distinct levels are present at which the objective function levels out for a few iterations and then continues to climb. In some cases this climb is dramatic, such as that in iterations 11 and 12, but in other cases it is gradual, such as iterations 20 through 23.

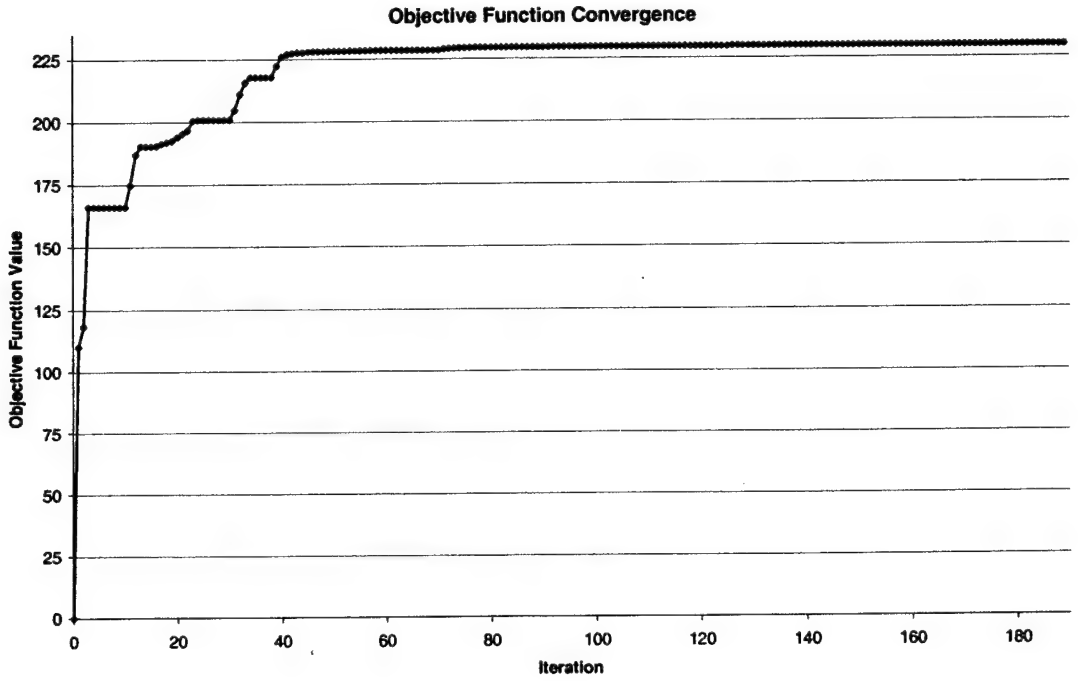

Figure 5-14: Objective function convergence of the LP for the full targeting cycle problem. We again see level portions of the graph and slow tail convergence.

Furthermore, we have the same constraining resources as previous discussed, but in this case the ISR assets are even more constraining as they are being used to perform BDA, jam antiaircraft threats, perform object discovery, and identify contacts. Contingency plans selected in this problem instance are similar to those discussed before, but with even more pauses waiting for resource availability due mostly to the large demand for ISR assets.

5.6 Chapter Summary

We have explored the LP/POMDP formulation, algorithm, and solutions to the targeting cycle problem. Many different structural variations and targeting cycle vignettes can be considered in exploring the behavior of this solution technique. φ and action control provide for

significantly faster solution times while attaining the same optimal solution as the uncontrolled problems. These improvements become important as we add higher dimensional POMDPs for object discovery and contact identification, which take a large amount of time to solve and to generate contingency plans. In this work we consider targeting cycle problems with 2 areas of interest and 3 contacts due to memory and computing power limitations. Larger problems with more areas of interest and contacts would better serve the operational community but would require fast machines with a large amount of memory or parallel processing of POMDP sub-problems. In general, however, the LP/POMDP formulation of the targeting cycle problem has much to offer the operational community such as the non-intuitive act-pause-act contingency plans and the ability to balance resources among a large number of targets, contacts, and areas of interest.

6 Summary and Future Work

This research has focused upon the application of optimization techniques to aid in the joint targeting cycle. Conventional modeling approaches have considered either resource allocation or policy development which do not adequately address the complexities of the targeting cycle problem. These shortcomings can be addressed by expanding the LP/POMDP framework proposed by Yost [37] to include object discovery, contact identification, and an intelligent adversary. This chapter serves as a summary of the work presented in this thesis, and we present suggestions for future research.

6.1 Thesis Summary

We introduced modeling and algorithmic changes to an existing methodology to improve the realism and computational aspects of the model and the associated solution algorithm. We enhanced this approach by 1) **incorporating the discovery and identification of targets**, 2) **handling regenerative targets**, and 3) **accounting for an intelligent adversary**. These aspects of the targeting cycle problem have not been developed in earlier works and this thesis represents the first piece of work addressing these issues. In addition to a more realistic model, we also enhanced the solution algorithm by proposing a **new initialization technique** as well as **two integer-programming formulations**. We ran experiments based upon a basic scenario, structural variations upon that scenario, and expanded targeting vignettes and investigated computational and qualitative characteristics of these solutions.

Chapter 1 lays the foundation and motivation for this research. We describe Air Operations Centers and the part they have to play in planning the air portion of a war. The battlefield is an ever-changing environment in which military planners must be able to respond quickly to enemy actions and provide the decision makers with plans to achieve military objectives. This is done through the targeting cycle, an implementation of Boyd's OODA loop.

In **Chapter 2**, we describe the targeting cycle the military uses to find, locate, and destroy the assets of an opposing military. We define the scope of this research to be the creation and selection of contingency plans with which to accomplish these three tasks using a limited amount of aircraft, weapons, and sensors. In this chapter, we discuss, in depth, the pre-strike ISR, strike, and post-strike ISR components of the targeting cycle. We also discuss the interactions between these phases which necessitates an integrated planning process.

In **Chapter 3**, we present the two primary modeling methods that have been applied to the targeting cycle problem: *resource allocation* and *policy development*. Resource allocation methods, namely mathematical programming formulations, are able to deal with large, complex planning problems but are not tractable with the large number of contingency plans possible in the targeting cycle problem. Policy development methods focus on resource costs rather than resource constraints and thus are unable to directly solve the targeting cycle problem. This suggests the use of column generation with a master level linear programming problem and policy development sub-problems, in this case, partially observable Markov decision processes. Therefore, we further explain the POMDP model and its solution techniques. Finally, we present the POMDP formulations for object discovery in an area of interest, contact identification, and targeting of identified contacts.

Using these POMDPs, we formulate the complete targeting cycle problem in **Chapter 4**. We also present a new initialization technique for the algorithm as well the assumptions we make about mobile contacts, regenerative targets, problem data, and executable integer solutions. Due to initial test results, and the findings of Yost [37], we also present two methods to speed up the POMDP sub-problems: φ*-control* and *action control*. Having provided the formulation, we discuss the techniques with which we will solve the targeting cycle problem. The Linear Support algorithm, as proposed by Cheng [13] is used to solve the two-state target POMDPs, while the Incremental Pruning algorithm [11] is used to solve area of interest and contact POMDPs. The master linear program is solved using the simplex method. We also discuss the interactions

between these two levels. While the dual information passed from the LP to the POMDPs is relatively straightforward, the construction of contingency plans based upon POMDP policies is much more difficult and thus is discussed at length.

We present scenarios solved with the LP/POMDP formulation in **Chapter 5**. These scenarios vary computational characteristics of the decomposition as well as the targeting cycle components that are included. We present the results of these tests as well as discuss some of the qualitative properties of the solutions. To aid the understanding of contingency plan construction, we provide a target POMDP policy and build three steps of the contingency plan associated with the policy.

In conclusion, the purpose of this thesis is to demonstrate the ability of an LP/POMDP formulation, and its associated solution algorithm, to aid in the targeting cycle process. While solution times for the complete targeting cycle problem are somewhat higher than we would like, they are still much faster and less labor intensive than the current methods employed in AOCs. Also, this optimization approach helps account for the system-wide impact of decisions made about individual objects leading to a globally optimal solution rather than a conglomeration of locally optimal solutions. This phenomenon is also present in the fact that modeling the complete targeting cycle problem yields higher value than a combination of models of the individual aspects of the targeting cycle. However, human interaction is needed if this approach were to be implemented in an operational setting. Post-implementation evaluation of selected contingency plans by human operators can improve the solutions provided.

6.2 Future Work

In this section we provide suggests for future research in applying the LP/POMDP formulation to the targeting cycle problem.

Movement Between Areas of Interest

Due to our independence assumption, movement of objects between areas of interest is not considered. These movements are a realistic part of the modern battlefield and can provide valuable information about military concentrations and enemy intent.

Discoveries as Observation Inputs for Contingency Plan Generation

As described in *Chapter 3, Section 3.1*, we do not consider the number of discoveries in an area of interest a separate indicator of the actual number of objects in that area. Inclusion of

discoveries as an observation would better model an intelligent adversary, one who would group military assets for mutual protection, as well as account for errors in intelligence preparation of the battlefield data.

Parallel Solutions to POMDPs

Looking at the results in *Chapter 5*, we see that the largest portion of time is spent solving the POMDP sub-problems. These problems, however, are independent of each other. The connection between them is made in the resource allocations of the master LP. Thus, once the master LP has been solved and the duals calculated, the POMDP sub-problems can be solved in parallel. This will reduce solution times while providing the same level of optimality. Especially of interest is solving the higher dimensional area of interest and contact POMDPs, which require long solution times compared to the two dimensional target POMDPs.

Approximate Solutions for Area of Interest and Contact POMDPs

As discussed in *Chapter 4*, finding optimal solutions to the POMDP sub-problems is not as crucial in the initial iterations of the LP/POMDP algorithm as it is in the later iterations. This, in conjunction with the fact that the area of interest and contact POMDPs take a long time to solve suggest a method similar to φ*-control* for those POMDPs. As proposed by Yost (1998) we could use grid-based POMDP solution algorithms to solve such POMDPs in the preliminary iterations of the algorithm. After switching criteria have been met, we would then solve these POMDPs with incremental pruning or another exact algorithm. Implementation of a grid-based method and the associated switching criterion could provide improvement in solution times thus allowing the LP/POMDP formulation to be used in real-time planning.

Comparison with other Techniques to Solve the Targeting Cycle Problem

While the solutions to the targeting cycle problem generated by the LP/POMDP formulation are valuable, contrasting this approach with other techniques would provide valuable insight into the strengths and weaknesses of each method. Metrics such as value attained and solution time are of interest as are the qualitative characteristics of the solutions generated.

Visualizations of Solutions

An important extension of this work would be to provide intuitive visualizations of the solutions generated, to include representations for contact contingency plans and resource usage. Such work would allow battlefield commanders to better utilize this optimization technology.

Appendix A: Formulations

This appendix serves as a reference for the five formulations used in this thesis. We begin with the LP formulations and then list the three POMDP formulations.

A.1 Set Definitions and Common Data

Aircraft types: $a \in A$
Weapons types: $w \in W$
ISR sensor types: $b \in B$
Object set: $i \in I$
Area of interest set: $\mathcal{A} \subseteq \mathrm{I}$
Contact set: $\mathcal{U} \subseteq \mathrm{I}$
Target set: $\mathcal{T} \subseteq \mathrm{I}$
Admissible contingency plans for object i: $o \in O_i$
Contact i possible types: $\xi \in \Xi_i$
States: $s \in S$
Allowable actions for object i : $\psi \in \Psi_i$
Possible observations for object i: $\theta \in \Theta_i$
Horizon: T
Time period: $t \in \mathbb{Z}^+ \leq \mathrm{T}$
Epoch: $k \in \mathbb{Z}^+ \leq T$

A.2 Master LP

- **Input Data**

  Average value of objects in area of interest i: EVAL_i
  Value of identifying contact i as type ξ given that it is of type ξ': $\mathrm{IDVAL}_{\xi\xi' i}$
  Value of target i: VAL_i
  Value of identifying contact i as type ξ given that it is of type ξ': $\mathrm{IDVAL}_{\xi\xi' i}$
  Number of aircraft of type a available at time t: NAA_{at}
  Number of ISR sensors of type b available at time t: ISR_{bt}
  Weapons availability of type w: WPN_w
  Maximum allowable attrition of aircraft type a: $\mathrm{MAXATTA}_a$
  Belief that contact i is of type ξ: $\mathrm{PT}_{\xi i}$
  Belief that target i is dead: PD_i

- **Contingency Plan Data**

  Expected number of aircraft type a needed to prosecute contingency plan o against object i in time period t: NA_{aoit}

Expected number of weapon type *w* needed to prosecute contingency plan *o* against object *i*: WE_{woi}

Expected number of ISR sensors *b* needed to prosecute contingency plan *o* against object *i* in time period *t*: LKS_{boit}

Expected attrition for aircraft type *a* under contingency plan *o* against object *i*: $ATTA_{aoi}$

Expected belief that target *i* is dead after applying contingency plan *o*: ED_{oi}

Probability of declaring contact *i* as type ξ after applying contingency plan *o*: $PDEC_{\xi oi}$

Expected number of discoveries in area of interest *i* under contingency plan *o*: $EDIS_{oi}$

- **Decision Variable**

  Apply contingency plan *o* to object *i*: x_{oi}

- **Objective Function**

$$\max_{x} \sum_{i \in \mathcal{A}} \sum_{o \in O_i} \mathrm{EVAL_i}\, \mathrm{EDIS_{oi}}\, x_{oi} + \sum_{i \in \mathcal{U}} \sum_{o \in O_i} \sum_{\xi \in \Xi_i} \sum_{\xi' \in \Xi_i} \mathrm{PDEC_{\xi oi}}\, \mathrm{IDVAL_{\xi\xi'i}}\, \mathrm{PT_{\xi' i}}\, x_{oi} +$$

$$\sum_{i \in T} \sum_{o \in O_i} \mathrm{VAL_i} \left(\mathrm{ED_{oi}} - \mathrm{PD_i} \right) x_{oi}$$

- **Constraints {Dual Information}**

$$\sum_{i \in I} \sum_{o \in O_i} NA_{aoit} x_{oi} \le NAA_{at} \quad \forall a,t \qquad \{\mathrm{ad_{at}}\}$$

$$\sum_{i \in I} \sum_{o \in O_i} WE_{woi} x_{oi} \le WPN_{w} \quad \forall w \qquad \{\mathrm{wd_w}\}$$

$$\sum_{i \in I} \sum_{o \in O_i} LKS_{boit} x_{oi} \le ISR_{bt} \quad \forall b,t \qquad \{\mathrm{ld_{bt}}\}$$

$$\sum_{i \in I} \sum_{o \in O_i} ATTA_{ao} x_{oi} \le MAXATTA_{a} \quad \forall a \qquad \{\mathrm{am_a}\}$$

$$\sum_{o \in O_i} x_{oi} = 1 \quad \forall i \qquad \{\mathrm{td_i}\}$$

$$0 \le x_{oi} \le 1 \quad \forall i \in I, o \in O_i$$

A.3 Dual Initialization LP

- **Input Data**

  Current belief that area *i* is in state *s*: PN_{si}

  Weighting factor between strike and BDA actions for targets: λ

- **Decision Variables**

  Proportion of action ψ to apply to object *i*: $x_{\psi i}$

- **Objective Function**

$$\max_{x} \sum_{i\in A}\sum_{\psi\in\Psi_i}\sum_{s\in S} \mathrm{EVAL_i}\ \mathrm{PDIS_i^{\psi}}\ \mathrm{PN_{si}}\,|s|\,x_{\psi i} +$$

$$\sum_{i\in A}\sum_{\psi\in\Psi_i}\sum_{\xi\in\Xi_i}\sum_{\xi'\in\Xi_i} \mathrm{PT_{\xi j}}\ \mathcal{O}_{\xi'}^{\psi\xi}\ \mathrm{IDVAL}_{\xi\xi' i}\ x_{\psi i} +$$

$$\sum_{i\in T}\sum_{\psi\in\Psi_i} \mathrm{VAL_i}\left[\left(\lambda[1-\mathrm{PD_i}]\mathcal{E}_{LD}^{\psi}\right)+(1-\lambda)\left([1-PD_i]\mathcal{O}_L^{\psi\,L}+\mathrm{PD_i}\,\mathcal{O}_D^{\psi\,D}\right)\right]x_{\psi i}$$

- **Constraints {Dual Information}**

$$\sum_{i\in I}\sum_{\psi\in\Psi_i} AFTUSE_a^{\psi}\,x_{\psi i} \le NAA_{a1} \quad \forall a \qquad \{\mathrm{ad_{at}}\}$$

$$\sum_{i\in I}\sum_{\psi\in\Psi_i} WPNUSE_w^{\psi}\,x_{\psi i} \le WPN_w \quad \forall w \qquad \{\mathrm{wd_w}\}$$

$$\sum_{i\in I}\sum_{\psi\in\Psi_i} SENSUSE_b^{\psi}\,x_{\psi i} \le ISR_{b1} \quad \forall b \qquad \{\mathrm{ld_{bt}}\}$$

$$\sum_{i\in I}\sum_{\psi\in\Psi_i} PA_{ai}^{\psi}\,x_{\psi i} \le MAXATTA_a \quad \forall a \qquad \{\mathrm{am_a}\}$$

$$\sum_{\psi\in\Psi_i} x_{\psi i} = 1 \quad \forall i \qquad \{\mathrm{td_i}\}$$

$$0 \le x_{\psi i} \le 1 \quad \forall i \in I,\ \psi \in \Psi_i$$

A.4 POMDP Models

A.4.1 Area of Interest POMDP Input Data

S={0, 1, 2, ...}

Θ={0, 1, 2, ...}

Probability action ψ will discovery an object when applied to area i: $\mathrm{PDIS_i^{\psi}}$

Probability of attrition of aircraft type a under action ψ applied to area i: $\mathrm{PA_{ai}^{\psi}}$

Average value of objects in area i: $\mathrm{AVGVAL_i}$

Cost of action ψ at epoch k applied to area i: C_{ik}^{ψ}

It is important to note that the discovery probability, $\mathrm{PD_i^{\psi}}$, is for each object in area i, independent of the other objects in area i.

A.4.2 Contact POMDP Input Data

$S=\Xi_i$

$\Theta=\Xi_i$

Probability contact *i* will evade when action ψ is applied: EV_i^{ψ}

Probability of receiving observation θ when action ψ is applied to contact *i* when it is in state *s*: $\mathcal{O}_s^{\psi\theta}$

Probability of attrition of aircraft type *a* under action ψ applied to contact *i*: PA_{ai}^{ψ}

Value of possible type ξ for contact *i*: AVGVAL_i^{ξ}

Cost of action ψ at epoch *k* applied to contact *i*: C_{ik}^{ψ}

If a contact evades, it is assumed to move to the "Not a target" state and thus is essentially lost. It is assumed that the contact will not evade if a "Pause" action is taken. The action list for contacts includes "Declare" actions that indicate a level of certainty that the contact is of a certain type. It is from these "Declare" actions that value is attained. Further explanation of the "Declare" actions can be found in *Chapter 4, Section 4.2.*

A.4.3 Target POMDP Input Data

S={Live, Dead}

Θ={Live, Dead}

Probability of kill for action ψ applied to target *i*: $\mathcal{E}_{LD}^{\psi}$

Probability of receiving observation θ when action ψ is applied to target *i* when it is in state *s*: $\mathcal{O}_s^{\psi\theta}$

Probability of attrition of aircraft type *a* under action ψ applied to target *i*: PA_{ai}^{ψ}

Probability of target *i* moving from dead state to live state due to repair when action ψ is applied: $\mathcal{E}_{DL}^{\psi}$

Value of target *i*: VAL_i

Cost of action ψ at epoch *k* applied to target *i*: C_{ik}^{v}

The repair probability, $\mathcal{E}_{DL}^{\psi}$, is assumed to be zero for relocatable targets.

Appendix B: Linear Support and Incremental Pruning Algorithms

In *Chapter 3, Section 3.3.2* we gave brief descriptions of several POMDP solution algorithms. Our use of both Linear Support and Incremental Pruning algorithms requires an understanding of the mechanisms behind these algorithms and the idiosyncrasies of implementing them. While this description takes an implementation slant, the reader is referred to Cheng [13], Zhang and Liu [38], and Cassandra, Littman, and Zhang [11] for full theoretical development of these two algorithms.

Both algorithms follow the general, finite horizon framework as shown in *Figure B-1*. What distinguishes them is the method by which the dynamic programming update is performed. The Linear Support algorithm is a *constructive method*, while Incremental Pruning is a hybrid between the *constructive* and *enumerative methods*.

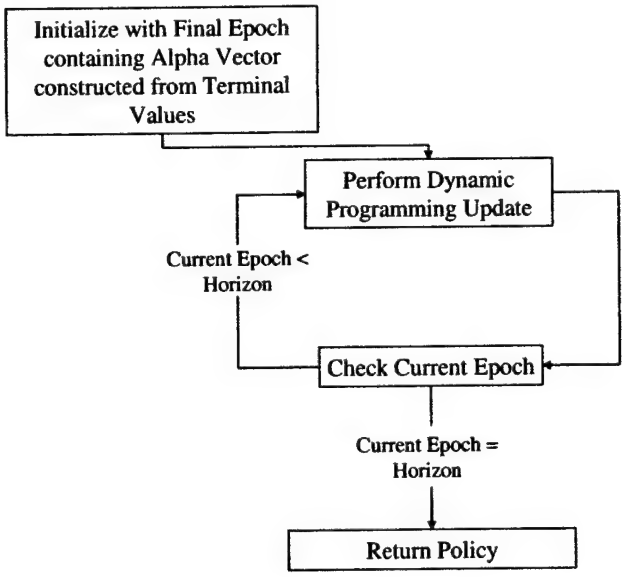

Figure B-1: General, finite horizon, POMDP algorithm framework

B.1 Linear Support Algorithm

To explain the linear support algorithm, a simple, two-state example is used. We assume that the following data is given: a value function over the belief space π for the previous epoch

$\left(V_{k-1}(\pi)\right)$ and a set of allowable actions for object i ($\psi \subseteq \Psi_i$) with associated costs/rewards (r^{ψ}), transition probabilities ($\mathcal{E}_{ss'}^{\psi}$), and observation probabilities ($\mathcal{O}_s^{\psi\theta}$). We suppress the i in the final three quantities, as well as for S, the set of states for object i, for ease of notation. Further, we define $\mathcal{P}$ to be the set of extreme points that need to be checked. Using the previous epoch and the action set, we will construct a value function for epoch k. This is done via the following algorithm that is illustrated in *Figure B -2*.

1. **Initialize $\mathcal{P}$ with the extreme points of the belief space.**
2. **Find the alpha vectors for the points in $\mathcal{P}$, store and mark them in $V_k(\pi)$, and clear $\mathcal{P}$.**
3. **If there are no marked alpha vectors in $V_k(\pi)$, go to 6.**
4. **Selected a marked alpha vector from $V_k(\pi)$. Find the selected alpha vector's extreme points at which it intersects |S|-1 other alpha vectors and add those points to $\mathcal{P}$.**
5. **Check the error between the true value function and the current approximation for the points in $\mathcal{P}$. Generate the optimal alpha vectors for the points with an error larger than φ. Add and mark them in $V_k(\pi)$. Unmark current alpha vector and clear $\mathcal{P}$. Go to 3.**
6. **$V_k(\pi)$ is complete value function for epoch k with maximum error less than or equal to φ.**

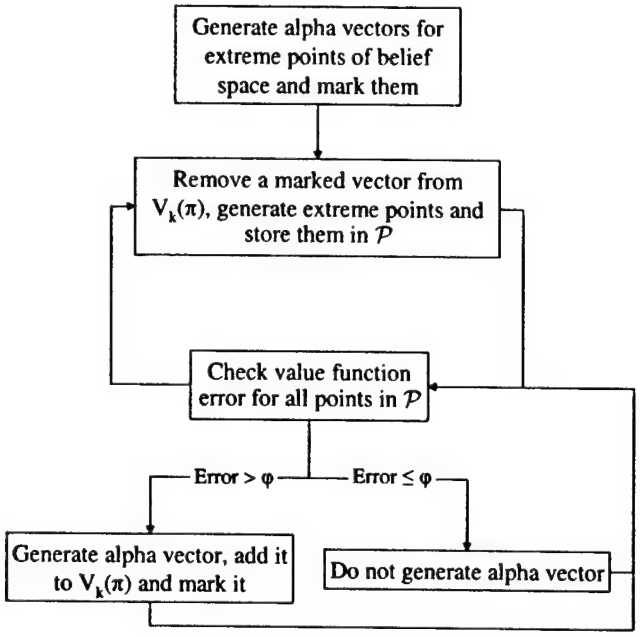

Figure B -2: Linear Support Algorithm DP update

This algorithm can be split up into three procedures: extreme point enumeration, alpha vector generation, and error checking.

B.1.1 Extreme Point Enumeration

While easy conceptually, extreme point enumeration is difficult in higher dimensions. In a two state problem, with a two dimensional value function, we simply find the two alpha vectors that were generated for belief states that most closely encompass the belief state for which the selected alpha vector was generated. Finding intersections of the selected alpha vector, α_i, with each of these alpha vectors is trivial and can be found using the following equation in which $\pi_{\text{intersect}}$ is a scalar because we are dealing with a two state problem.

$$\pi_{\text{intersect}} = \frac{\alpha_j(k,0) - \alpha_i(k,0)}{\alpha_i(k,1) - \alpha_i(k,0) - \alpha_j(k,1) + \alpha_j(k,0)}. \tag{B.1}$$

This process becomes much more difficult in three dimensions. A closed form equation such as (B.1) could be derived for finding the intersection of three alpha vectors. However, determining which two alpha vectors to use in conjunction with α_i is not inconsequential. It thus becomes necessary to use an algorithm from computational geometry such as the ones described by Mattheiss [28] or Mattheiss and Rubin [29]. These algorithms search over a defined polyhedron and return all of the extreme points. To do this, most algorithms set up and solve a fairly complex linear programming problem. Polynomial time deterministic algorithms are available to solve such a problem [8]. The necessity to do such calculations, however, somewhat diminishes the original intention for the Linear Support algorithm "to develop an algorithm which does not require complicated constraint sets" [13].

B.1.2 Error Checking

Once we have found the extreme points that need to be considered for α_i we must check the error between the true value function and the current approximation. This should be done even if we are implementing linear support as an exact algorithm so that we do not expend computational resources generating the optimal alpha vector for a belief state at which we already have the optimal alpha vector. To find the error between the optimal value, $val_k^*(\pi)$,

and the current approximation, $val_k(\pi)$, at a belief state π we consider the difference between (B.2) and (B.3).

$$val_k^*(\pi) = \max_{\psi \in \Psi_i} \left\{ \sum_{s \in S} r_{sk}^{\psi} \pi(s) + \sum_{\theta \in \Theta_i} \left[\max_{u \in U(k-1)} \sum_{s \in S} \left(\sum_{s' \in S} \mathcal{E}_{s's}^{\psi} \pi(s') \right) \mathcal{O}_s^{\psi\theta} \alpha_u(k-1, s) \right] \right\}. \quad \text{(B.2)}$$

$$val_k(\pi) = \max_{u \in U(k)} \{ \alpha_u \cdot \pi \}. \quad \text{(B.3)}$$

If this difference is less than or equal to the error tolerance, φ, then this point can be ignored. However, if the calculated error is larger than the error tolerance, then we must generate the optimal alpha vector for the belief state $\pi_{\text{intersect}}$.

B.1.3 Alpha Vector Generation

As described in *Chapter 3, Section 3.3.2.1*, we use (B.4) and (B.5) to generate an alpha vector for a given belief state π and action ψ:

$$\alpha^{\psi}(k, s) = r_{sk}^{\psi} + \sum_{\theta \in \Theta_i, s' \in S} \mathcal{E}_{ss'}^{\psi} \mathcal{O}_{s'}^{\psi\theta} \alpha_{\kappa(\pi,\psi,\theta)}(k-1, s'). \quad \text{(B.4)}$$

$$\kappa(\pi, \psi, \theta) = \arg\max_{u \in U(k-1)} \sum_{i,j \in S} \pi(i) \mathcal{E}_{ij}^{\psi} \mathcal{O}_j^{\psi\theta} \alpha_u(k-1, j). \quad \text{(B.5)}$$

The first of these equations combines the current value and the expected future value. The future value comes from the appropriate future alpha vectors as chosen by (B.5). (B.5) returns the index for the alpha vector we would use in the next time step if we were at belief state π, took action ψ, and received observation θ. We do not receive the full value from this alpha vector though. Rather, the value must be weighted by the probability that the object transitions from s to s' and then we receive the observation θ. To find the optimal alpha vector for the belief state $\pi_{\text{intersect}}$, $\alpha_{\pi}^{\psi^*}(k)$, we use equation (B.4) to generate an alpha vector based upon the maximizing action from (B.2). Along with the end point values from equation (B.4), we also store $\pi_{\text{intersect}}$ and the action associated with the new alpha vector.

B.1.4 Graphical Example of a Linear Support Algorithm DP Update for a 2 State Problem

The first step of the DP update is to find the extreme points of the belief space and then generate alpha vectors for these points, *Figure B-3*. Select either of these alpha vectors and

check its extreme points. One has already been checked and the other is the intersection of the two alpha vectors, *Figure B-4*. We find this intersection and check the difference between the optimal value and the current approximate value, *Figure B-5*. In this case, the error is larger than φ and we use equations (B.4) and (B.5) to generate a new alpha vector, *Figure B-6*, for that belief state, storing it in $V_k(\pi)$, *Figure B-7*. We then select the other alpha vector generated for the end points of the belief space. Since its extreme points have already been checked we select $\alpha_3^{w_3}(k)$ and find its extreme points, *Figure B-8*.

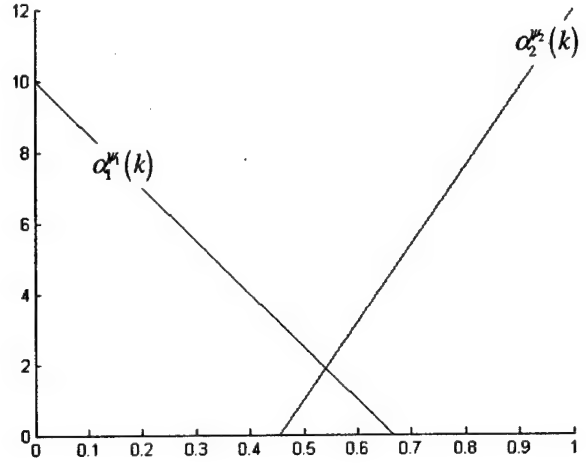

Figure B-3: Generation of Alpha Vectors for Extreme Points of Belief Space

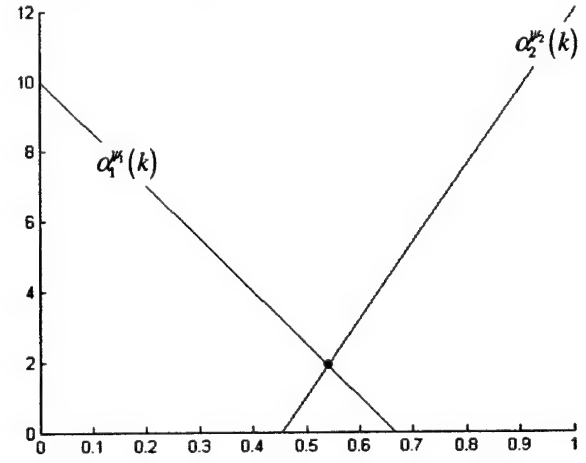

Figure B-4: Extreme Point Enumeration for Current Alpha Vector

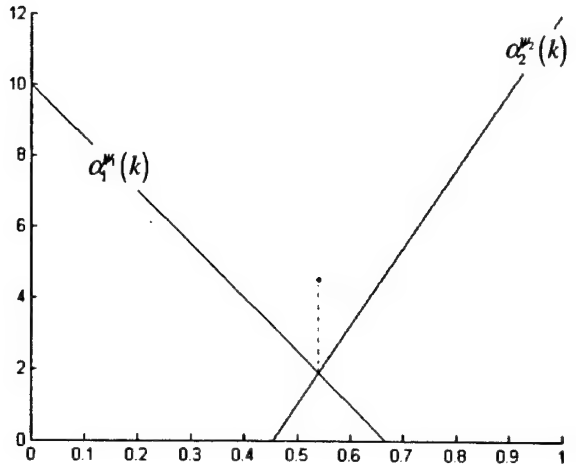

Figure B-5: Calculation of Error at Extreme Point

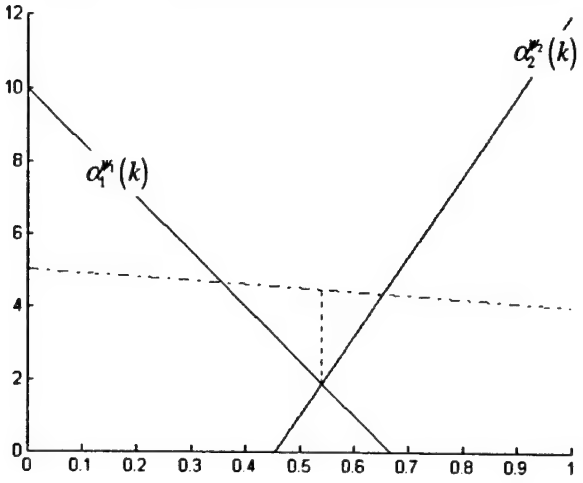

Figure B-6: Generation of Alpha Vector at Extreme Point

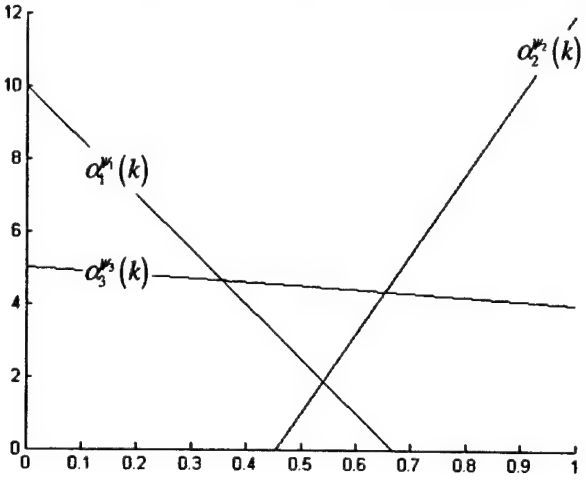

Figure B-7: Updated Approximation of Value Function

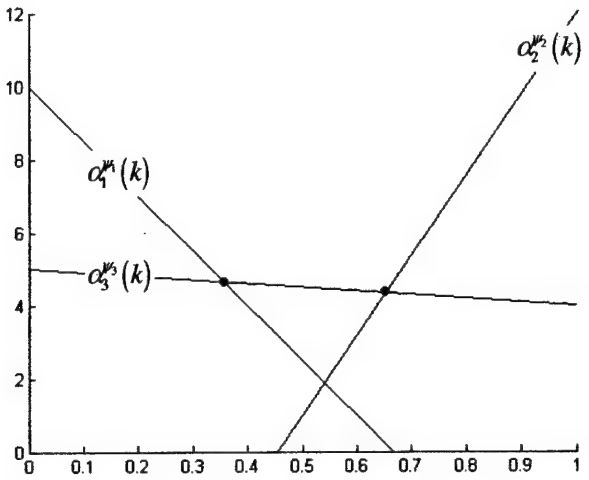

Figure B-8: Extreme Point Enumeration for Current Alpha Vector

Once we have found these extreme points, we can check the value function error. We first check $\alpha_3^{w_3}(k)$'s right hand extreme point, *Figure B-9*, and find that the error is larger than φ therefore we need to generate the optimal alpha vector associated with that point, *Figure B-10.* Again, we use (B.4) and (B.5) based upon the maximizing action from (B.2). The generated alpha vector is then stored in $V_k(\pi)$ and marked for later consideration. We then check the other extreme point of $\alpha_3^{w_3}(k)$, *Figure B-11.* In this case, the value function error is less than φ thus we do not generate the optimal alpha vector for that point.

We then move on to the next marked vector, which was the fourth alpha vector we generated, $\alpha_4^{w_4}(k)$. In finding the endpoints and checking the value function error at these points, *Figure B-12*, we see that both of the errors are less than φ and so no additional alpha vectors are generated. There are no more marked alpha vectors and our DP updates is done. We have found the φ-optimal value function, *Figure B-13.* This value function specifies the regions of the belief space over which different actions are optimal. That is, the range over which each alpha vector in the value function dominates, is the range of belief points for which its associated action is optimal.

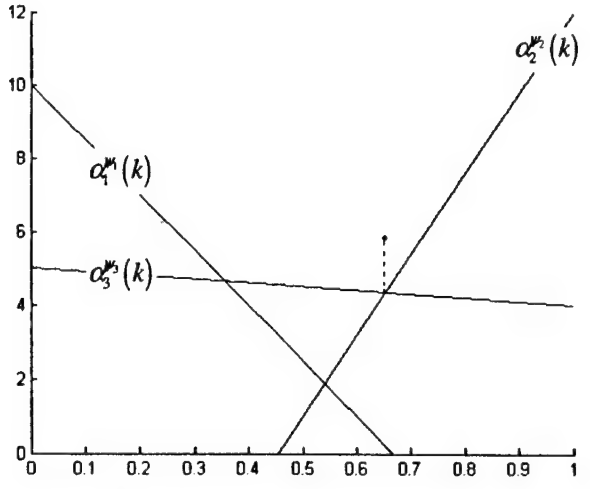

Figure B-9: Calculation of Error at Extreme Point

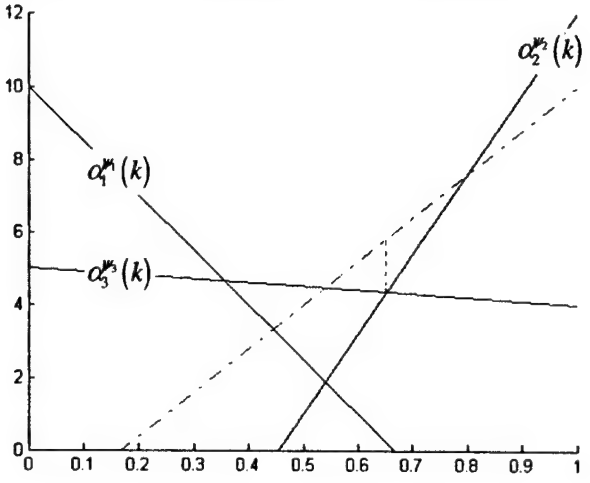

Figure B-10: Generation of Alpha Vector at Extreme Point

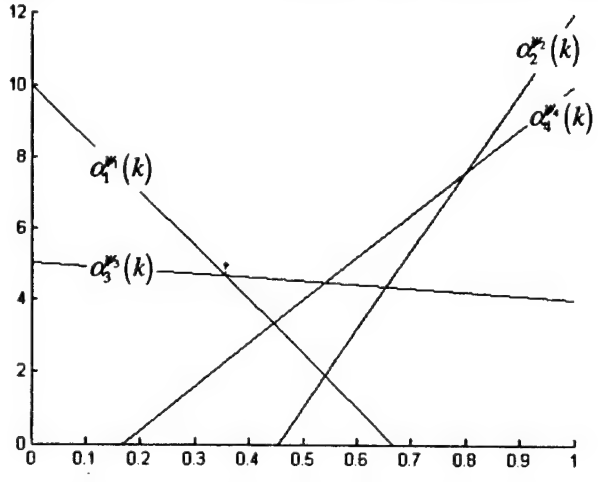

Figure B-11: Calculation of Error at Extreme Point

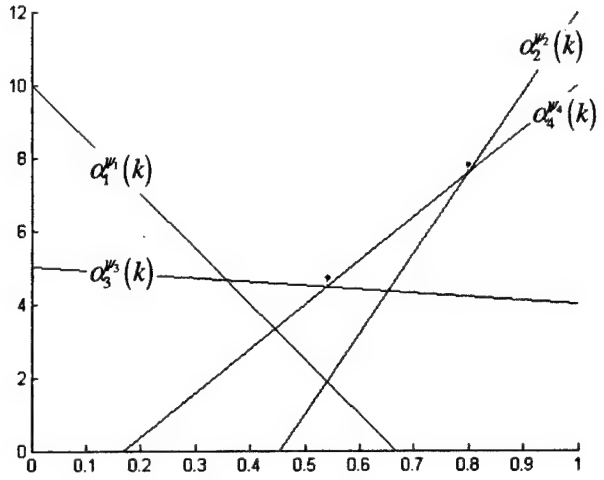

Figure B-12: Calculation of Error at Extreme Points

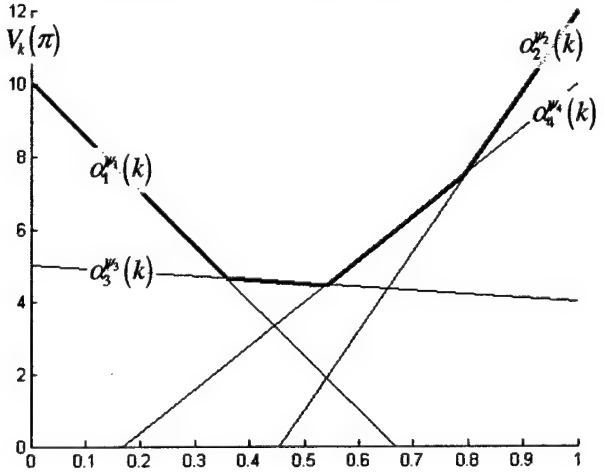

Figure B-13: Final φ-Optimal Value Function

B.2 Incremental Pruning Algorithm

While the Linear Support algorithm is intuitive and easily demonstrated for a two state problem, implementation in higher dimensions is not straightforward. With the possible number of intersections increasing exponentially with the size of the state space, the computational effort needed to implement the algorithm will greatly increase. A method that combines some ideas from the enumerative algorithms and some from the constructive algorithms would better serve us when dealing with higher dimensional POMDPs. While we lose the ability to maintain an approximate solution at every iteration of the algorithm, as is possible in linear support, this will be offset by the increased efficiency of the algorithm.

Zhang and Liu presented such an algorithm called Incremental Pruning [38]. The key to this algorithm is that when (B.6) is used for a dynamic programming update of the value function, it must be done for all possible combinations of actions, observations, and future alpha vectors.

$$\alpha^{\psi}(k,s) = r_{sk}^{\psi} + \sum_{s' \in S, \theta \in \Theta} \mathcal{E}_{ss'}^{\psi} \mathcal{O}_{s'}^{\psi\theta} \alpha_u(k-1,s') \quad \forall \psi \in \Psi, \theta \in \Theta, u \in U. \tag{B.6}$$

Zhang and Liu showed that sets of alpha vectors could be generated based upon a fixed action and observation for all future alpha vectors using the following equation:

$$\frac{1}{|\Theta|} r_s^{\psi} + \sum_{s' \in S} \mathcal{E}_{ss'}^{\psi} \mathcal{O}_{ss'\theta}^{\psi} \alpha_u(k-1,s'). \tag{B.7}$$

These alpha vectors could then be compared and dominated alpha vectors removed thus returning the parsimonious set, that is the set of dominate alpha vectors, for the given action and observation, $V_k^{\psi\theta}(\pi)$. Once all of the parsimonious sets for a given action-observation pair have been found, they can be incrementally combined and extraneous alpha vectors removed, yielding a parsimonious set for the action, $V_k^{\psi}(\pi)$. For all the possible actions, these sets are then combined and the parsimonious set found; this set is the optimal set of alpha vectors for the current epoch, $V_k(\pi)$. The full algorithm is as follows with *Figure B-14* providing a graphical representation.

1. **Set ψ equal to first action in Ψ_1 and θ to first observation in Θ.**
2. **Generate alpha vectors for all $\alpha_u(k-1)$ using ψ, θ, and equation (B.7). Filter these alpha vectors and set $V_k^{\psi\theta}(\pi)$ equal to the result.**
3. **If θ is the last observation in Θ go to 4, else increment θ and go to 2.**
4. **Set $V_k^{\psi}(\pi)$ equal to incrementally pruned $V_k^{\psi\theta}(\pi)$ sets.**
5. **If ψ is the last action in Ψ go to 6, else increment ψ and go to 2.**
6. **Filter $\bigcup_{\psi \in \Psi_1} V_k^{\psi}(\pi)$ and set $V_k(\pi)$ equal to the result.**

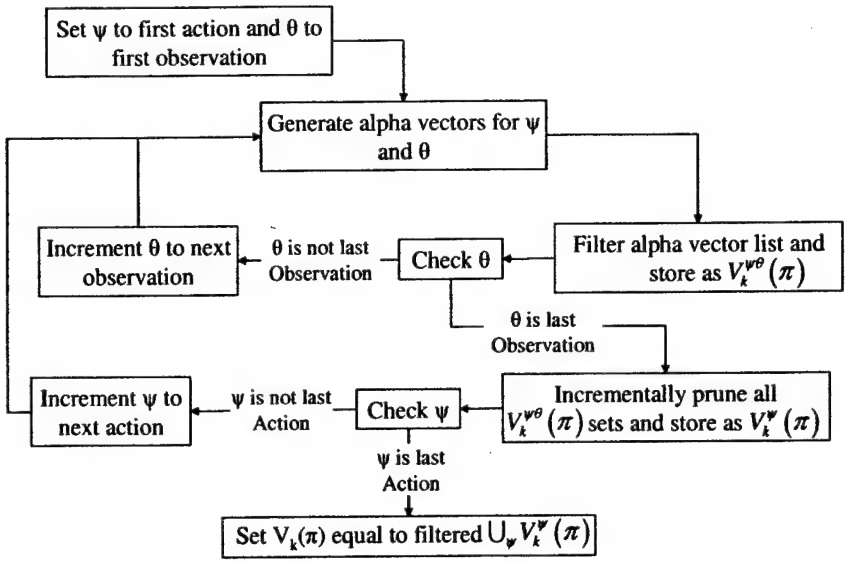

Figure B-14: Incremental Pruning Algorithm DP update

As in the linear support algorithm, the procedures of this algorithm can be split up. In this case the partitions are the filter and incremental pruning steps.

B.2.1 Filter

The basic idea of the filter operations is to reduce a set of alpha vectors to its parsimonious set. This is done in a similar way to Monahan's algorithm with Eagle's modification as described in *Chapter 3, Section 3.3.2.1.* We begin with a set of alpha vectors, $\mathcal{S}$, which might contain dominated alpha vectors. We want to reduce $\mathcal{S}$ to a set $\mathcal{V}$ which only contains alpha vectors that dominate over a convex neighborhood within the belief space. This is done using the following algorithm in which $\mathcal{V} \subseteq \mathcal{S}$ at all times:

1. **Set *s* to first state in S.**
2. **Find the alpha vector in $\mathcal{S}$ that dominates at *s* and add it to $\mathcal{V}$ if it is not already in $\mathcal{V}$.**
3. **If *s* is last state in S, go to 4, else increment *s* and go to 2.**
4. **Remove all alpha vectors from $\mathcal{S}$ that are now in $\mathcal{V}$.**
5. **Pick an alpha vector α^* from $\mathcal{S}$.**
6. **Perform *Dominate Function*, see *Section B.2.1.1*, on α^* and $\mathcal{V}$ setting π to returned belief state. If π is null, remove α^* from $\mathcal{S}$ and go to 8.**
7. **Find alpha vector that dominates at π, remove it from $\mathcal{S}$ and add it to $\mathcal{V}$.**
8. **If $\mathcal{S}$ does not equal the null set, go to 5. Else, $\mathcal{V}$ is parsimonious set.**

B.2.1.1 Dominate Function

An added function is necessary to complete the filter algorithm. This is the dominate function. The dominate function sets up and solves an LP to find a belief state at which we need to find the dominant alpha vector. The dominate function is spelled out below for inputs α^* and $\mathcal{V}$.

1. **Define an LP with decision variables π_s and δ and objective function to maximize δ.**
2. **Pick first alpha vector in $\mathcal{V}$ and define it as α.**
3. **Define a set of constraints such that $\sum_{s\in S}\pi_s\left(a(s)-a^*(s)\right)\leq -\delta$.**
4. **If α was last alpha vector in $\mathcal{V}$ go to 5, else set α as next alpha vector in $\mathcal{V}$ and go to 3.**
5. **Add the constraint $\sum_{s\in S}\pi_s = 1$.**
6. **Add nonnegativity constraints for all π_s.**
7. **Check LP for infeasibility. If it is infeasible, return null.**
8. **Solve LP. If δ>0 return decision variable values. If δ=0, return null.**

The dominate command sets up and solves the following formulation:

$$\max_{\pi} \quad \delta \tag{B.8}$$

$$\sum_{s\in S}\pi_s\left(\alpha(s)-\alpha^*(s)\right)\leq -\delta \qquad \forall \alpha \neq \alpha^* \tag{B.9}$$

$$\sum_{s\in S}\pi_s = 1 \tag{B.10}$$

$$\pi_s \geq 0 \qquad \forall s\in S. \tag{B.11}$$

As you can see, this is exactly Monahan's reduction algorithm, as discussed in *Chapter 3, Section 3.3.2.1* with an objective function that ensures that chosen alpha vectors dominate over a convex neighborhood of the belief space.

B.2.2 Incremental Pruning

While the first and last combinations of alpha vector sets, $V_k^{\psi\theta}(\pi)$ and $\bigcup_{\psi\in\Psi_i} V_k^{\psi}(\pi)$, in the Incremental Pruning algorithm are accomplished much like Monahan's algorithm with Eagle's modification, the middle combination is quite different. It is the key to the Incremental Pruning algorithm and its efficiency. The key observation here is that when we combine the sets

of $V_k^{\psi\theta}(\pi)$ for different θs, we can combine the first two and then remove any dominated alpha vectors. We can then combine this parsimonious set with the next $V_k^{\psi\theta}(\pi)$ and then purge that set. This is continued until all of the $V_k^{\psi\theta}(\pi)$ sets have been combined and pruned. This incremental combining and pruning gives the algorithm its name.

To do this, remember how the initial alpha vectors were generated. The immediate costs/rewards were scaled by the number of possible observations while the future rewards were scaled by the probability of receiving the given observation. Thus, to combine the sets we must sum the entire first set of alpha vectors with all of the alpha vectors from the second set. It is this *cross-summed* set from which we will remove dominated alpha vectors. Each $V_k^{\psi\theta}(\pi)$ is cross summed with the current parsimonious set and then purged of dominated alpha vectors. The incremental pruning step is:

1. **Select $V_k^{\psi\theta_1}(\pi)$ and $V_k^{\psi\theta_2}(\pi)$.**
2. **Cross sum two selected vector sets and set equal to $\mathcal{S}$.**
3. **Filter $\mathcal{S}$ and set result to $\mathcal{V}$.**
4. **If there is another $V_k^{\psi\theta}(\pi)$ select it and $\mathcal{V}$ and go to 2. If not, go to 5.**
5. **Set $V_k^{\psi}(\pi)$ equal to $\mathcal{V}$.**

Cassandra, Littman, and Zhang [11] explore a modification to this algorithm that change how the filter and dominate functions work during the incremental pruning step. Instead of using the alpha vectors from $\mathcal{V}$ when constructing the constraints for the LP, they use cross-summed alpha vectors from the two sets being considered. They also show other vector sets that could be used in place of $\mathcal{V}$.

B.3 Summary

Linear Support and Incremental Pruning both provide benefits when solving certain types of POMDPs. With no theoretical proof as to which algorithm has better general characteristics we use both algorithms in solving our POMDP problems. Linear support is used for the target POMDPs because we model targets as having two states, live and dead, and the ability to generate φ-optimal value functions. Incremental pruning, on the other hand, is used for the area of interest and contact POMDPs due to the varying and potentially large size of the state space.

[This Page Intentionally Left Blank]

Appendix C: Glossary of Acronyms

ACC..... Air Component Commander

AGM..... Air to Ground Missile

AI..... Artificial Intelligence

AOC ..... Air Operations Center

BDA ..... Battle Damage Assessment

C2 ..... Command and Control

CAOC..... Combined Air Operations Center

CBU..... Cluster Bomb Unit

BLU..... Bomb Live Unit

DP..... Dynamic Program

EW..... Early Warning

F..... Military Designation for Fighter Aircraft

GPS..... Global Positioning System

GBU ..... Guided Bomb Unit

GHZ..... Gigahertz

HQ ..... Headquarters

IAD..... Integrated Air Defense

IP ..... Integer Program

IPB..... Intelligence Preparation of the Battlefield

ISR..... Intelligence, Surveillance, and Reconnaissance

JFACC..... Joint Forces Air Component Commander

LOAC..... Law of Armed Conflict

LP ..... Linear Program

MB..... Megabytes

MDP ..... Markov Decision Process

MEA..... Munitions Effectiveness Assessment

MIP..... Mixed Integer Program

OODA ..... Observation, Orientation, Decision, Action

OR ..... Operations Research

PMF..... Probability Mass Function

POMDP..... Partially Observable Markov Decision Process

RQ..... Military Designation for Reconnaissance Drones

RAM..... Random Access Memory

RR..... Restricted Region

SAM..... Surface-to-Air Missile

SSM..... Surface-to-Surface Missile

UAV..... Unmanned Aerial Vehicle

Appendix D: Notation

Sets and Common Data

$a \in \mathrm{A}$	Aircraft resources
$b \in \mathrm{B}$	Sensor resources
$w \in \mathrm{W}$	Weapon resources
$i \in \mathrm{I}$	Object set
$\mathcal{A} \subseteq \mathrm{I}$	Area of interest set
$\mathcal{U} \subseteq \mathrm{I}$	Contact set
$\mathcal{T} \subseteq \mathrm{I}$	Target set
$j \in \mathrm{J}$	Generic resource set
$o \in \mathrm{O_i}$	Contingency plans for object i
$s \in \mathrm{S}$	Object states
$\psi \in \Psi_i$	Allowable actions for object i
$\xi \in \Xi_i$	Contact i possible types
$\theta \in \Theta_i$	Possible observations for object i
T	Horizon
ε	Numeric tolerance used to determine if two floating point numbers are equal
$n \in \mathbb{Z}^+ \leq T$	Time step
$k \in \mathbb{Z}^+ \leq T$	Epoch
$u \in U(k)$	Index of Alpha Vectors for epoch k
$\phi \in \Phi_{ot}$	Set of belief state nodes for contingency plan o at time step t

Linear Programming

$\mathcal{R}_{oi}$	Reward for using contingency plan o against object i
$\mathcal{U}_{joi}$	Resources of type j used by contingency plan o against object i
$\mathcal{Y}_j$	Resources of type j available
x_{oi}	Variable representing the decision to use a proportion of contingency plan o against object i
$\mathcal{C}_T$	Number of contingency plans for problem with horizon of T
O_i^{ψ}	Contingency plan for object i that has an initial action of ψ

$x_{\psi i}$	Variable representing the decision to use a proportion of action ψ against object i
x_{oi}^{ψ}	Variable representing the decision to use a proportion of contingency plan o, which has an initial action of ψ, against object i
y_i^{ψ}	Binary variable for the decision to use contingency plans with an initial action ψ against object i
$\mathbf{p}$	Column of dual values from a linear programming problem
$\mathbf{c_B}$	Objective function coefficients for the variables in the optimal solution of a linear programming problem
$\mathbf{B}$	Matrix composed of the columns of the variables in the optimal solution of a linear programming problem

POMDP

s_t	State of an object at time t
S_{max}	Maximum number of objects in an area of interest
ψ_t	Action taken at time t
$\mathcal{E}_{ss'}^{\psi}$	Probability that an object transitions to state s' from state s when action ψ is taken
$\mathcal{O}_{s'}^{\psi\theta}$	Probability of observing θ given that the object is in state s and action ψ was taken
r_{sk}^{ψ}	Reward for taking action ψ when in state s at epoch k
π	PMF over possible states
π_s	Decision variable in domination check LP
$\pi_{\text{intersect}}$	Belief state representing the intersection of alpha vectors
λ	Resource costs
$\pi(s)$	Probability that an object is in state s
r_{s0}	Terminal value for being in state s
$\alpha^{\psi}(k)$	Alpha vector associated with action ψ in epoch k
$\alpha^{\psi}(k,s)$	Value of alpha vector at state s with action ψ in epoch k
$\alpha_u^{\psi}(k)$	Alpha vector with index u with associated action ψ in epoch k
$\alpha_u^{\psi}(k,s)$	Value of alpha vector with index u at state s with action ψ in epoch k
$V_k(\pi)$	Value function for epoch k over belief space represented by π

$\kappa(\pi,\psi,\theta)$	Function used to determine alpha vector for use in update equation
φ	Allowable error in value function when solving with linear support algorithm
$V_k^{\psi\theta}(\pi)$	Value function for epoch k based upon action ψ and observation θ
$V_k^{\psi}(\pi)$	Value function for epoch k based upon action ψ
C_{ik}^{ψ}	Cost of action ψ at epoch k applied to object i
$\mathcal{P}$	List of extreme points to check under linear support algorithm
$val_k^*(\pi)$	Optimal value at belief point π at epoch k
$val_k(\pi)$	Value at belief point π at epoch k based upon approximate value function
I_k	Information vector for k steps
$\psi_k^*(I_k)$	Optimal action for information vector I_k
π_k	Sufficient statistic for k steps, sufficient statistic is the belief state for POMDPs
$\psi_k^*(\pi_k)$	Optimal action for sufficient statistic π_k
$\alpha^*(k)$	Alpha vector under consideration in domination checks
$\alpha_\pi^{\psi^*}(k)$	Optimal alpha vector for belief state π with associated optimal action ψ^*
$\mathrm{EV}_i^{\psi}(s)$	State dependent evasion probability for a contact
$\mathcal{Y}_\pi^{\psi\theta}$	Probability of occurrence for a belief state node with an associated observation θ who's parent node had a belief state of π and optimal action of ψ
$\mathcal{Y}_P$	Probability of occurrence for a generic parent belief state node
$\mathcal{Y}_\phi$	Probability of occurrence for belief state node ϕ
$E\pi_{it}$	Expected belief state for target i at time step t
$\mathcal{S}$	A set of alpha vectors that may contain non-dominate alpha vectors
$\mathcal{V}$	A set of alpha vectors that dominate over a region of the belief space
δ	Variable in dominate function of incremental pruning forcing a neighborhood dominance for alpha vectors added to $\mathcal{V}$

[This Page Intentionally Left Blank]

References

[1] Air Force Fact Sheet. "AGM-65 Maverick. " October 1999. <http://www.af.mil/news/factsheets/AGM_65_Maverick.html> (Feburary 26, 2003).

[2] Air Force Fact Sheet. "Global Hawk. " December 2002. <http://www.af.mil/news/factsheets/global.html> (Feburary 25, 2003).

[3] Air Force Fact Sheet. "RQ-1 Predator Unmanned Aerial Vehicle. " May 2002. <http://www.af.mil/news/factsheets/RQ_1_Predator_Unmanned_Aerial.html> (Feburary 25, 2003).

[4] Air Force Pamphlet 14-210. *USAF Intelligence Targeting Guide*. February 1, 1998.

[5] Air University Database. "Cluster Bombs. " <http://www.au.af.mil/au/database/projects/ay1996/acsc/96-004/hardware/docs/cluster.htm> (Feburary 27, 2003).

[6] American Forces Information Service. "The Operation Desert Shield/Desert Storm Timeline." August 08, 2000. <http://www.defenselink.mil/news/Aug2000/n08082000_20008088.html> (February 28, 2003).

[7] Bertsekas, Dimitri P. *Dymnamic Programming and Optimal Control*. Belmont, MA: Athena Scientific. 2000.

[8] Bertsimas, Dimitris, and Tsitsiklis, John N. *Introduction to Linear Optimization*. Belmont, MA: Athena Scientific. 1997.

[9] Birge, John R., and Louveaux, Francois. *Introduction to Stochastic Programming*. New York: Springer-Verlag New York, Inc., 1997.

[10] Cassandra, Anthony R. "Optimal policies for partially observable Markov decision processes." Technical report CS94-14, Brown University. Providence, Rhode Island, 1994.

[11] Cassandra, Anthony, Littman, Michael L., and Zhang, Nevin L. "Incremental Pruning: A Simple, Fast, Exact Method for Partially Observable Markov Decision Processes." Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence, 1997.

[12] Castañon, David A. "Approximate Dynamic Programming for Sensor Management." Proceedings of the 36th IEEE Conference on Decision and Control, Vol. 2, IEEE Control Systems Society, Danvers, MA, pp 1202-1207, 1997.

[13] Cheng, Hsien-Te. *Algorithms for Partially Observable Markov Decision Process*. Ph.D. thesis, University of British Columbia, Vancouver, British Columbia, 1988.

[14] Clausewitz, Carl Von. *On War*. Princeton: Princeton University Press. 1990.

[15] Dantzig, George B. *Linear Programming and Extensions*. Princeton: Princeton University Press, 1963.

[16] Drake, A. W. *Observation of a Markov Process Through a Noisy Channel*. PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1962.

[17] Dyer, M. E., "The complexity of vertex enumeration methods." *Mathematics of Operations Research*, Vol. 8, pp. 381-402, 1983.

[18] Eagle, James N. "The Optimal Search for a Moving Target when the Search Path is Constrainted." *Operations Research*, Vol. 32, No. 5, pp. 1107-1115, 1984.

[19] Elliott, Scott, Master Sgt. "Air Force Rethinks Air Operations Centers." February 26, 2003. <http://www.af.mil/stories/22603562.shtml> (February 26, 2003).

[20] Gilmore, P. C. , and Gomory, R. E. "A linear programming approach to the cutting stock problem". *Operations Research*, Vol. 9 pp. 848-859, 1961.

[21] Gilmore, P. C. , and Gomory, R. E. "A linear programming approach to the cutting stock problem-Part II". *Operations Research*, Vol. 11, pp 863-888, 1963.

[22] Hauskrecht, Milos. *Planning and Control in Stochastic Domains with Imperfect Information*. Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1998.

[23] Hillier, Frederick S., and Lieberman, Gerald J. *Introduction to Operations Research*. New York: McGraw-Hill, Inc., 1995.

[24] Joint Publication 3-60. *Joint Doctrine for Targeting*. January 17, 2002.

[25] Littman, Michael L. "The witness algorithm: Solving partially observable Markov decision processes." Technical report CS94-40, Brown University. Providence, RI, 1994.

[26] Littman, Michael L., Dean, Thomas L., and Kaelbling, Leslie Pack. "On the complexity of solving Markov decision problems." In Proc. of the Eleventh International Conference on Uncertainty in Artificial Intelligence, 1995.

[27] Lovejoy, William S. "A Survey of Algorithmic Methods for Partially Observed Markov Decision Processes." *Annals of Operations Research*, Vol. 28, No. 1, pp. 47-65, 1991.

[28] Mattheiss, T. H. "An Algorithm for Determining Irrelevant Constraints and all Verticies in Systems of Linear Inequalities." *Operations Research*, Vol. 21, pp. 247-260, 1973.

[29] Mattheiss, T. H., and Rubin, D. S. "A Survey and Comparison of Methods for Finding All Verticies of Convex Polyheral Sets." *Mathematics of Operations Research*, Vol. 5, pp. 167-185, 1980.

[30] Meuleau, Nicolas, Hauskrecht, Milos, Kim, Kee-Eung, Peshkin, Leonid, Kaelbling, Leslie Pack, Dean, Thomas L., and Boutilier, Craig, "Solving Very Large Weakly Coupled Markov Decision Processes." Proceedings of the Conference on Uncertainty in Artificial Intelligence, 1998.

[31] Monahan, George E., On Optimal Stopping in a Partially Observable Markov Chain with Costly Information. Ph.D. Disseration, Northwestern University, 1977.

[32] Murkejee, Sraban, and Seth, Kiran. "A Corrected and Improved Computational Scheme for Finite Horizon Partially Observable Markov Decision Process." *INFOR*, Vol. 29, No. 3, pp. 206-212, 1991.

[33] Papadimitriou, Christos H., and Tsitsiklis, John N. "The Complexity of Markov Decision Processes." *Mathematics of Operations Research*, Vol. 12, pp. 441-450, 1987.

[34] Sondik, Edward J.. The optimal control of partially observable Markov processes. Ph.D. Thesis, Stanford, 1971.

[35] United States. General Accounting Office. *Operation Desert Storm: Evaluation of the Air Campaign*. Washington: GPO, 1997.

[36] Wikipedia Online Encyclopedia. "Computational complexity theory." <http://www.wikipedia.org/wiki/Computational_complexity_theory> (April 7, 2003).

[37] Yost, Kirk A. *Soluiton of Large-Scale Allocation Problems with Partially Observable Outcomes*. Ph.D. Dissertation in Operations Research. Naval Postgraduate School, Monterey, California, September 1998.

[38] Zhang, Nevin L., and Liu, Wenju. "Planning in stochastic domains: Problem characteristics and approximation." Technical report HKUST-CS96-31, Department of Computer Science, Hong Kong University of Science and Technology, Kowloon, Hong Kong, 1996.

[39] Zhou, R., and Hansen, E. "An improved grid-based approximation algorithm for POMDPs." In Proceedings of 17th International Joint Conference on Artificial Intelligence, 1998.

[This Page Intentionally Left Blank]